

SCELBI's
GALAXY
GAME
FOR THE
"6800"



**SCELBI COMPUTER
CONSULTING INC.**

SCELBI'S GALAXY GAME

FOR THE '6800'

TABLE OF CONTENTS

	Introduction
Chapter ONE	Operation of the Galaxy Program
Chapter TWO	System requirements
Chapter THREE	Data Table, Messages and Subroutines
Chapter FOUR	Major Routines of Galaxy
Chapter FIVE	'6800' Assembler Listing
Chapter SIX	Sample of Galaxy Operation

INTRODUCTION

Imagine yourself as captain of a space ship traveling throughout the galaxy. Your mission is to seek and destroy all alien ships to make the galaxy safe so that other ships from your planet may journey into outer space. Due to the urgency of the mission it must be completed within a given time. If the mission is not completed within the time allotted, the safety of all future voyages is in jeopardy. Your space ship is supplied with a limited amount of fuel and weapons so you must choose your course and attack strategy carefully. Mission control has placed space stations at various points in the galaxy for refueling. A space station contains a limitless amount of fuel and weapons. However, don't get caught too far from a space station with your energy low or you may end up drifting endlessly through space.

As an aid in searching the galaxy, the space ship is equipped with a galaxy scanner which is capable of displaying three different degrees of detail. The short range scan provides an accurate picture of the immediate quadrant through which the space ship is currently traveling. Your location and that of any alien ships, stars, and space stations in the quadrant are defined by exact sector coordinates. The long range scan displays the contents of the eight quadrants surrounding the quadrant you presently reside in. The wide angle scanner provides a view of the total galaxy from which you can plot your course.

The space ship is equipped with two types of weapons. The PHASOR is an energy discharge device which homes in on all enemy ships in the immediate area and directs specified amounts of energy at each. This energy, if enough to destroy, will completely eliminate the alien ship. However, should the alien ship survive the attack, it will retaliate by shooting back at your ship. It is important that you keep the energy in your ship's protective shields at sufficient levels to withstand any possible retaliation from the enemy. The other weapon available is the TORPEDO. It is capable of destroying any alien ship on impact. The target must be in direct line of sight of the space ship for the torpedo to reach its destination. A missed tor-

pedo shot results in immediate retaliation by the alien ship. Also, be careful when there is a space station in the area. If the torpedo hits it, the space station is destroyed.

Now, turn your imagination into the realm of reality by transforming your small computer system into the control station of the space ship. Each move by the space ship is controlled by the computer operator and the responsibility of the total mission is placed on the operator's shoulders. The GALAXY program presented here will allow one to make this transformation by loading the program as presented, and simply adding the appropriate I/O routines for one's specific I/O setup. Or, it can be expanded by revising the command operations or adding new commands to make the game more complex, and modifying it to take advantage of special I/O devices which the reader may have associated with one's computer system. The number of possible variations are limitless. The operation of this program is explained in detail to aid those that desire to make revisions and additions to its operation.

OPERATION OF THE GALAXY PROGRAM

Before getting into the specifics of the SCLEBI GALAXY program, it is important that the reader understands the general operation of the program. As one might imagine, the programming will be a bit intricate at times, so a good general knowledge of its operation will help keep things in perspective. This section is also written so that it may be used as an operating guide which may be referred to when playing the game.

The object of the Galaxy game is to destroy all the alien ships in the galaxy. The exact number of alien ships which must be destroyed is defined in the initial message along with the number of stardates one has to complete the mission, and the number of space stations available in the galaxy for refueling. Each time a game is started, the entire galaxy is set up in a random manner so that no two games will be the same. The number of alien ships and space stations, and their respective locations in the galaxy will also be different for each game.

The galaxy is made up of 64 quadrants arranged in an eight-by-eight matrix. The quadrants are identified by the row number and column number of its location in the matrix. The row numbers run from one to eight starting with the top row. The column numbers go from one to eight starting with the left-hand column. Within each quadrant there are 64 sectors arranged in the exact same format as the quadrants in the galaxy. There can exist only one galactic object in a sector at any one time. An illustration of the matrix is shown on the following page.

The space ship used to traverse the galaxy in search of enemy vessels contains several integral parts which allow it to carry out its mission. First, there is the main storage bank which contains the main supply of energy for the space ship. This energy is used to move the ship through the galaxy, supply the power to fire the phasors and torpedoes, and transfer energy to the protective shields. The maximum energy capacity in the main storage bank is 5000 units.

1 2 3 4 5 6 7 8

1								
2								
3								
4								
5								
6								
7								
8								

The master control panel is used to enter commands to direct the ship's movement, request scanner displays, fire phasors and torpedoes, and transfer energy to the protective shields. It also displays status reports to inform the operator of various conditions which arise during the course of the mission. The master control panel requires 10 units of energy for each command entered. It is also a positive action panel which means that once a command mode is entered, the command sequence must be completed. The physical arrangement of the master control panel will depend on the I/O facilities of the individual computer system.

The alien ships which are to be destroyed have the following properties. First, a protective shield, similar to the space ship's shields, surrounds the alien ship. This shield can contain from 0 to 1023 units of energy. This supply of energy is depleted by a phasor shot from the space ship in direct proportion to the amount of phasor energy which reaches the shield. Next, the alien ship has an endless supply of energy to fire back at the space ship. This energy is fired only in retaliation for an attack by the space ship. If a torpedo shot misses, the alien ship responds with a phasor of 200 units of

energy. If a phasor does not destroy the alien ship, a phasor with 1/4 of the amount of energy left in the shields of the alien ship is fired at the space ship. The alien ship is destroyed by the direct hit of a torpedo, by a phasor which removes all of its shield energy, or by the space ship colliding with the alien ship.

Space stations are scattered throughout the galaxy to provide the space ship with refueling locations. In order for the space ship to refuel, it must maneuver to a sector alongside the space station where it is considered "docked." When the space ship is docked, its energy supply is replenished to its maximum capacity, and the torpedo tubes are refilled to their capacity of 10 torpedoes. The energy and torpedoes are transferred to the space ship only on the initial move to dock with the space station. Remaining docked while using energy to fire phasors and torpedoes will not provide the space ship with an endless supply. To replenish its supply after attacking from a docked position, the space ship must move away from, and then return to, the space station. Also, when docking, if the space ship collides with the space station, the space station will be destroyed.

The ship's weapons arsenal consists of a phasor, which discharges high levels of concentrated energy, and a torpedo launcher. The phasor "homes in" on all alien ships in the quadrant in which the space ship is residing. The actual amount of energy fired is selected by the operator. The torpedo must proceed in a straight line to the object that it is to destroy. The maximum number of torpedoes, and the amount of energy used for each, will be covered shortly.

The protective shields are the ship's defense against any attack by an alien ship, or its protection from damage should it accidentally collide with a space station or alien ship. The shields are capable of absorbing an amount of energy equal to the amount of energy they contain. It is important that the shield energy level be maintained high enough to withstand any possible attack, since severe energy losses occur if the shield energy goes to zero.

The stars, which are scattered throughout the galaxy, serve as possible obstructions for the space ship when moving about in a quadrant, and by blocking the direct line of fire of a torpedo. The space ship must also be very careful in maneuvering around a star

because colliding with one means instant destruction.

When a command is to be input to the program, the following message will be displayed:

COMMAND?

The operator must then enter a number from zero to six to initiate one of the following command modes.

- 0 - SPACE SHIP MOVEMENT COMMAND
- 1 - SHORT RANGE SCAN COMMAND
- 2 - LONG RANGE SCAN COMMAND
- 3 - GALAXY DISPLAY COMMAND
- 4 - SHIELD COMMAND
- 5 - PHASOR COMMAND
- 6 - TORPEDO COMMAND

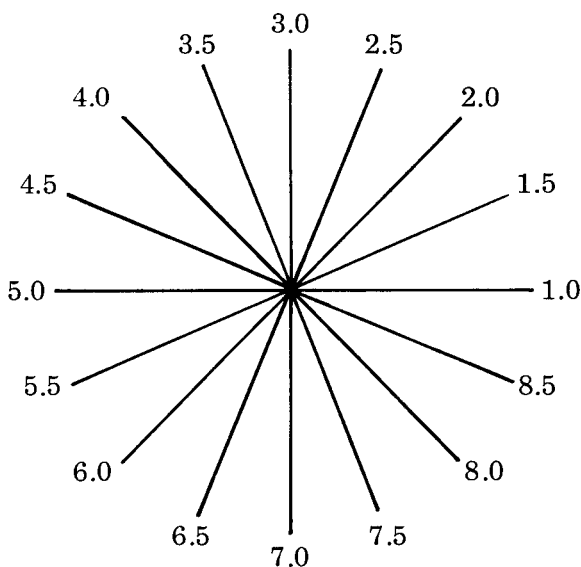
COMMAND 0 - SPACE SHIP MOVEMENT

The movement of the space ship is controlled by designating both course direction and distance. Movement within a quadrant requires only the energy required for the command, which is 10 units. If the move crosses one or more quadrant boundaries, 25 units are used for each quadrant crossed. When the completion of any move results in the space ship residing in a new quadrant, one stardate is used up.

When a movement command is entered, the course direction is requested by the following message being displayed:

COURSE (1-8.5)?

Course direction is entered by specifying a two digit number as indicated in the request of the value 1.0 to 8.5. This number indicates the direction the space ship is to move according to the compass on the following page.



From this diagram, one can see that the possible directions start to the right with a value of 1.0 and move around in a counterclockwise manner with assignments made every $22\frac{1}{2}$ degrees. If one desired to move to the left and slightly down, the course would be entered as 5.5. This direction assignment is also used to define the trajectory of a torpedo fired from the space ship, as will be discussed shortly.

After the direction has been entered, the distance, or warp factor, is requested by the following message being displayed:

WARP FACTOR (0.1-7.7)?

As indicated, the warp factor is entered by specifying a two digit value. The space ship will move a distance of one sector for each 0.1 designated in the input. The maximum value for either digit is seven. Thus, to move to the same sector in the quadrant to the right of one's current position, the course direction would be 1.0, and the warp factor would be 1.0, not 0.8. This setup creates a one-to-one

relationship between the distance entered, and the number of quadrants and sectors moved through, since the quadrants are broken up into an 8 x 8 matrix for the sectors.

There are several moves which one must be very careful to avoid while traveling through the galaxy. One is that of moving out of the boundaries of the galaxy. If this occurs, the space ship is lost forever in outer space. Another move of equivalent consequence is a move which causes the space ship to crash into a star. A star is considerably larger than the space ship, and a collision results in the space ship becoming completely engulfed in the gaseous composition of the star and destroyed. The third move to avoid is a collision with a space station. The force of the collision will result in the loss of 600 units of energy from the space ship. Of a greater consequence, however, is the aspect that the space station is wiped out on impact, since it contains no defensive mechanism to absorb such a collision. This may seriously damage the chances of completing a mission. The final move is a "kami-kazi" move against an alien ship. This gives the desired affect of destroying the enemy, but the space ship will sustain a loss of 1500 units of energy which may leave it with very little power. The possibility of colliding with another object is only present while traveling in the quadrant that the space ship is in at the time the movement command was entered. Once the ship moves outside the initial quadrant, the automatic guidance control takes over and safely steers the space ship to its destination.

COMMAND 1 - SHORT RANGE SCAN

The short range scan provides a detailed picture of the contents of the quadrant in which the space ship currently resides. A short range scan uses only the energy required for the command, which is 10 units. The precise sector location of the space ship, stars, alien ships, and space stations are displayed for examination by the operator. The following symbols are used to define each of the possible objects.

<*> - SPACE SHIP
+++ - ALIEN SHIP
* - STAR
>1< - SPACE STATION

A sample of a short range scan display is shown below. The display also provides the basic status information for the ship to the left of the scan. The stardate will always start with a 30, and the last two digits will approach the value of 50. When the stardate reaches 3050, the space ship has run out of stardates and the mission has failed. The condition status will be red if an alien ship is present in the current quadrant, and green if there are no alien ships in the quadrant. The quadrant and sector values refer to the current position of the space ship. The first digit indicates the row number, and the second digit indicates the column of the respective position in the galaxy. The energy is the amount of energy currently contained in the main storage bank. This energy will be a maximum value of 5000 units. The next entry provides a count of the number of torpedoes available on the space ship. The final status entry indicates the amount of energy in the protective shields.

- 1 - - 2 - - 3 - - 4 - - 5 - - 6 - - 7 - - 8 -		
1		*
2		
3		+++
4	*	
5	<*>	
6		
7	>1<	*
8		
- 1 - - 2 - - 3 - - 4 - - 5 - - 6 - - 7 - - 8 -		

EXAMPLE OF A SHORT RANGE SCAN

COMMAND 2 - LONG RANGE SCAN

The long range scan command gives an overall view of the eight quadrants which surround the quadrant currently occupied by the space ship. The 10 units of energy needed for a command are all that is required to display a long range scan. The presence of alien ships, space stations and stars are indicated for each quadrant. The contents are indicated by a three digit number in each square. The left hand digit indicates the number of alien ships in the quadrant; the center

digit indicates the number of space stations, and the right hand digit indicates the number of stars. A sample of a long range scan is presented below.

L.R. SCAN FOR QUADRANT 6,5

1 112 1 001 1 006 1

1 001 1 113 1 104 1

1 203 1 007 1 004 1

COMMAND 3 - GALAXY DISPLAY

The contents of the entire galaxy may be displayed by requesting a galaxy display. The display requires only the 10 units of energy necessary for the command. The contents of each quadrant are shown in the same form as that used in the long range scan. From this display the operator may plan a long range course to successfully complete a mission. The following is a sample of a galaxy display. The reader may note the location of the long range scan quadrants as pre-

1 105 1 002 1 003 1 000 1 000 1 105 1 000 1 000 1

1 117 1 000 1 304 1 106 1 005 1 003 1 107 1 002 1

1 105 1 007 1 003 1 006 1 000 1 000 1 000 1 000 1

1 005 1 003 1 000 1 000 1 000 1 000 1 003 1 004 1

1 001 1 000 1 000 1 112 1 001 1 006 1 203 1 105 1

1 000 1 103 1 000 1 001 1 113 1 104 1 002 1 117 1

1 000 1 103 1 000 1 203 1 007 1 004 1 000 1 002 1

1 000 1 000 1 003 1 000 1 000 1 001 1 102 1 107 1

sented in the previous illustration.

COMMAND 4 - SHIELD CONTROL

The shield control command provides a means of transferring energy between the main energy storage bank and the protective shields. The shields must contain energy to protect the space ship from attacks by the alien ships or from possible collisions with either an alien ship or a space station. The energy required to make the transfer is simply the 10 units required for the command. The amount of energy transferred is specified by the operator in response to the following message being displayed:

SHIELD ENERGY TRANSFER =

The operator then enters a four digit number indicating the amount of energy desired to be transferred. When a four digit number is entered, the energy is transferred from the main storage bank to the shield. If a four digit number is preceded by a minus sign, the energy is transferred from the protective shield back to the main storage bank.

It is important that the amount of energy in the shields be maintained at sufficient levels to withstand any possible attack. If the shield energy should become too low to absorb the energy of an attack, the additional energy needed will be taken from the main supply, and an additional 25 percent of the total energy loss will be depleted from the main storage bank as a penalty. This 25 percent loss is the amount of energy required to make repairs to the portions of the space ship damaged by the energy that was not absorbed by the shields.

COMMAND 5 - PHASOR CONTROL

The phasor control directs the phasor's energy at the alien ships that reside in the immediate quadrant. The amount of energy that is to be fired is specified by the operator in response to the following message being displayed.

PHASOR ENERGY TO FIRE:

A four digit number is then entered and the phasor shots are fired at the alien ships in the quadrant. The result of the phasor energy shot at each alien ship is reported by the following message being displayed:

ALIEN SHIP AT SECTOR X,Y: ENERGY = ZZZZ
or DESTROYED

The values of X and Y indicate the sector location of the alien ship, and the message after the colon will indicate either the amount of energy (ZZZZ) remaining in the alien ship, or that the alien ship has been destroyed. If the alien ship is not destroyed by the phasor, one quarter of its energy will be shot back at the space ship in retaliation. This retaliation will be indicated by the following message:

LOSS OF ENERGY XXXX

Before specifying the amount of energy, the operator must be aware of several properties of phasor energy. First, the amount of energy to be fired is divided equally between the alien ships in the quadrant. If there are two alien ships in the quadrant, and the operator indicates 500 units of energy, 250 units will be fired at each alien ship. Next, the amount of phasor energy that reaches the target is governed by the distance the energy must travel. The distance is figured by adding up the number of sectors in the horizontal and vertical direction between the space ship and the alien ship. This distance is then divided by four and the fraction is discarded; this value is used as the distance factor. The distance factor is the number of times the amount of energy fired at an alien ship is to be divided by two. The distance between the space ship and the alien ship is therefore critical to the amount of phasor energy to reach the alien ship. For example, if the space ship is at sector 2,4 and the alien ship is at sector 6,6, the total number of sectors is equal to two in the horizontal direction ($6-4=2$) plus four in the vertical direction ($6-2=4$). This distance of six is divided by four and the whole number one is used as the distance factor. This distance factor divides the energy to be fired at the alien ship by 2. It is important that the space ship be as close to the alien ship(s) as possible to achieve the

maximum effectiveness of a phasor shot.

COMMAND 6 - TORPEDO CONTROL

The torpedo control fires a torpedo in the direction specified by the operator. Each torpedo requires 250 units of energy to fire, and must be in the direct line of fire of the target. The trajectory of the torpedo is entered by the operator in response to the following message being displayed:

TORPEDO TRAJECTORY:

The trajectory is defined in the same format as the course specification when entering a movement command. A two digit number is entered indicating the direction in which the torpedo is to travel. The track of the torpedo is then traced, and reported to the operator as it moves from one sector to another. This is reported by a series of tracking messages displayed in the following format:

TRACKING: X,Y
TRACKING: U,V
TRACKING: S,T

The values of X,Y, U,V, and S,T are the row and column of the sectors through which the torpedo is passing. When the torpedo either reaches the boundary of the quadrant or hits an object, an advisory message is displayed. If the torpedo misses the alien ship and reaches the boundary of the quadrant, or if the torpedo hits a star, the following message will be displayed:

YOU MISSED! ALIEN SHIP RETALIATES
LOSS OF ENERGY = 200

Missing the alien ship causes it to retaliate by firing back 200 units of energy at the space ship. If the torpedo hits a space station, not only is the alien ship going to retaliate, but the space station is destroyed since it has no defense against a torpedo. The following message is displayed to inform the operator of this serious disaster.

SPACE STATION DESTROYED
YOU MISSED! ALIEN SHIP RETALIATES
LOSS OF ENERGY = 200

If all goes well, and the trajectory is right on target, the alien ship will be destroyed and the following message will inform the operator of the successful hit:

ALIEN SHIP DESTROYED

SYSTEM REQUIREMENTS

MEMORY USAGE FOR THE GALAXY PROGRAM

The Galaxy program presented in this book requires 4096 bytes of RAM memory to operate in a 6800-based microcomputer system. The program is broken down into the following blocks of memory. Page 00 is used to store the course table, temporary data, the galaxy display line, and the galaxy content table. Pages 01 through 04 contain the messages used by the program. The subroutines reside on pages 05 through 09, and the major program routines run from the middle of page 09 to page 0E. The lower half of page 0F is used to store the galaxy setup table and the upper half of page 0F is reserved for the user supplied input and output routines. The stack is initialized at the top of page 0E. If more than 128 bytes are required by the user for the I/O routines, and the user's system does not have more than 4K of memory, the length of several of the messages can be cut down to provide the additional memory space needed for the I/O routines.

INPUT/OUTPUT REQUIREMENTS

The input/output requirements for the galaxy program presented herein allow the reader to tailor the I/O portion of the program to the specific devices that are available for use on one's computer system. The character code used in this program is the 7 bit ASCII code with the 8th bit, or parity bit, assumed to be at a '1.' The game uses the full alphanumeric character set plus punctuation marks. A table of the ASCII code required by this program is presented on the following page.

The input routine must input a character from the system input device, such as a keyboard, and return to the calling program with the ASCII code for the character entered in the A accumulator. This input routine, labeled INPUT, can use the A accumulator to input the character. If the B accumulator or the Index register must be used, the input routine must save and then restore their contents before returning. If the input device is not connected in some way

ASCII CHARACTER SET

CHARACTERS SYMBOLIZED	HEXA REP	CHARACTERS SYMBOLIZED	HEXA REP
A	C1	!	A1
B	C2	”	A2
C	C3		A3
D	C4	\$	A4
E	C5	%	A5
F	C6	&	A6
G	C7	,	A7
H	C8	(	A8
I	C9	)	A9
J	CA	*	AA
K	CB	+	AB
L	CC	,	AC
M	CD	-	AD
N	CE	.	AE
O	CF	/	AF
P	D0	0	B0
Q	D1	1	B1
R	D2	2	B2
S	D3	3	B3
T	D4	4	B4
U	D5	5	B5
V	D6	6	B6
W	D7	7	B7
X	D8	8	B8
Y	D9	9	B9
Z	DA	:	BA
[	DB	;	BB
	DC		BC
]	DD	=	BD
	DE		BE
	DF	?	BF
SPACE	A0	@	C0
CAR RET	8D	RUBOUT	FF
LINE FEED	8A		

to the display device to provide automatic printout of the characters entered, the INPUT routine should provide some means of outputting the character received to the output device. This may be achieved by echoing the character in the input routine, or by calling the PRINT routine to perform the output. The INPUT routine is called in the subroutines labeled DRCT and EIN, and in the major routines labeled GALAXY, CMND and CRSE.

The output routine is required to output the character whose ASCII code is contained in the A accumulator when the output routine is called. The initial contents of the A and B accumulators and the Index register are expected to be maintained upon returning to the calling program. If it is necessary to use these registers, the contents must be saved and then restored upon returning to the calling program. The output routine is referred to by the label PRINT. This routine is called by the subroutines MSG, NTN and DRCT, and the major routine CRSE.

For systems that operate with the MIKBUG** program for I/O, the following program listings may be used for the INPUT and PRINT routines.

INPUT	JSR \$E1AC ORAA #\$80 RTS	Call MIKBUG** input routine Set the parity bit Return to the calling program
PRINT	PSHA JSR \$E1D1 PULA RTS	Save character to be output Call MIKBUG** output routine Restore character in A Return to calling program

** MIKBUG is a registered trademark of the Motorola Corporation

DATA TABLE, MESSAGES, and SUBROUTINES

DESCRIPTION OF THE GALAXY DATA ON PAGE 00

The major portion of the operation of the Galaxy game concerns itself with the contents and manipulation of the data stored on page 00 from location 33 to 50. This table area is reserved for the storage of information, such as the location of the space ship, stars, alien ships, and space stations within the current quadrant, the amount of energy contained in the main energy storage, the shields of the space ship, and the energy in the shields of the alien ships. The count of the number of torpedoes, space stations, alien ships, and stardates remaining is also stored here. The format of the data in this table is summarized below with a description of each following the summary.

LOCATIONS	FORMAT	DEFINITION
33, 34	XXXXXXXX	Random number storage
35	00AASTTT	Current quadrant contents
36	00RRRCCC	Sector location of space ship
37, 3D	00RRRCCC	Sector location of stars
3E	00RRRCCC	Sector location of space station
3F	00RRRCCC	Sector location of alien ship No. 1
40	00RRRCCC	Sector location of alien ship No. 2
41	00RRRCCC	Sector location of alien ship No. 3
42, 43	XXXXXXXX	Dbl precision value of main energy
44, 45	XXXXXXXX	Dbl precision val. of shield energy
46, 47	XXXXXXXX	D.P. val. of alien ship No. 1 energy
48, 49	XXXXXXXX	D.P. val. of alien ship No. 2 energy
4A, 4B	XXXXXXXX	D.P. val. of alien ship No. 3 energy
4C	00RRRCCC	Crnt. quad. location of space ship
4D	0000PPPP	Number of torpedoes remaining
4E	00000XXX	Number of space stations
4F	000XXXXX	Number of alien ships
50	00XXXXXX	Number of stardates remaining

LOCATIONS 33 and 34

The random number routine uses the contents of these two locations to generate and store the next random number.

LOCATION 35

The contents of the current quadrant in which the space ship is located are stored in this byte. The bits indicated by TTT provide a count of the number of stars in the quadrant; the S indicates a space station present when set to "1;" and the bits AA indicate the number of alien ships in the quadrant. Each time a new quadrant is entered, this location is loaded with its contents. This is done to provide the program with a convenient reference location for the contents of the quadrant. All of the quadrants are set up at the start of the game, and stored in the galaxy content table on the upper quarter of page 00.

LOCATION 36

The row and column numbers for the current sector location of the space ship are indicated by the RRR and CCC bits, respectively, in this byte. The row and column numbers are represented by the binary values zero through seven in this location. However, they represent the row and column numbers one through eight when presented in the output to the display device. This row and column representation is used in the next 11 locations to indicate the location of the stars, space stations, and alien ships in the quadrant. This provides the program with a convenient means of checking for a strike by a torpedo, or a collision of the space ship with another object in the quadrant. The initial value stored in this location is set up using the random number generator. After that time, the location of the space ship is controlled by the operator.

LOCATIONS 37 through 3D

The location of the stars in the current quadrant is indicated by the row and column numbers contained in this portion of the table. The values RRR and CCC are of the same format as that presented for the space ship. If there are less than seven stars in the current quadrant, the unused locations in this table are set to octal C0. If there are no stars in the quadrant, all of the locations will contain

C0. The locations of the stars are set using the random number generator each time a new quadrant is entered.

LOCATION 3E

The location of the space station in the current quadrant is stored here. The row and column numbers are represented in the same format as the space ship and stars; they are set by use of the random number generator each time a new quadrant is entered. If a space station does not reside in the current quadrant, this location will contain C0. At the completion of a move by the space ship, this location is used in determining whether the space ship has docked with the space station.

LOCATIONS 3F through 41

This portion of the table is used for the storage of the location of the alien ships. The row and column representation is the same as that presented for the previous nine locations. If less than three alien ships are in the current quadrant, the unused locations will contain C0. When an alien ship is destroyed, the corresponding location in this table will be set to C0 as part of the process of eliminating it from the galaxy.

LOCATIONS 42 and 43

The binary value of the amount of energy in the main storage bank is maintained in this location pair. The least significant half is saved in location 42, and the most significant half in location 43. The maximum value stored in this location is 5000, which is set up at the start of a game and each time the space ship docks.

LOCATIONS 44 and 45

This location pair is used to store the binary value of the energy contained in the space ship's protective shields. As with the main energy storage, the least significant half is stored in location 44, and the most significant half in location 45. The amount of energy stored in this location is set up by a command entry, and is depleted by attacks by alien ships.

LOCATIONS 46 through 4B

The binary values of the energy levels of the alien ships protective shields are contained in this portion of the table. The least significant half is in the even numbered byte, and the most significant half in the odd numbered byte. The energy level for each alien ship is set up using the random number generator when a space ship enters a quadrant. The energy indicated in these locations is the only defense an alien ship has against a phasor attack.

LOCATION 4C

This location contains the row and column numbers of the space ship's current quadrant location within the galaxy. The format is the same as that for the sector location of the space ship defined previously. The quadrant location is set up initially by use of the random number generator, and is then controlled by the operator as the space ship is moved throughout the galaxy. The contents of this location are used to fetch the quadrant contents by setting the two most significant bits to "1," and using this as a pointer to the galaxy content table.

LOCATION 4D

A count of the number of torpedoes remaining in the space ship is maintained here. This location is set to 10 at the start of each game and each time the space ship docks with the space station. When a torpedo is fired, this count is decremented by one until it reaches zero which indicates there are no torpedoes left.

LOCATION 4E

This location maintains a count of the number of space stations in the galaxy. If a space station is destroyed by collision or torpedo, the count is decremented by one. When the count goes to zero, a warning message is displayed to inform the operator that the last space station has been destroyed.

LOCATION 4F

A count of the number of alien ships remaining is maintained in this location. Each time an alien ship is destroyed, this location is de-

cremented by one. When it reaches zero, the mission is completed by the successful destruction of all the alien ships.

LOCATION 50

This location indicates the number of stardates left in the game. A stardate is used up when a move results in the space ship residing in a new quadrant. This location will be decremented by one each time this occurs. When this count goes to zero, the operator has run out of time, and the game is over. This count is initially set to five more than the number of alien ships.

The data area on page 00 is followed by a list of pointers. This table of pointers begins at location 56 and runs up through location 75 on page 00. The pointers contained in this table point to various locations in the data table that the index register is set to at different times throughout the program. By setting up a table in this fashion, the index register may be loaded by a two-byte direct addressing mode instruction rather than a three-byte extended addressing mode instruction, thereby shortening the program by approximately one quarter of a page. This table of pointers is presented in the assembled listing in Chapter Five.

TEXT MESSAGES USED IN THE GALAXY PROGRAM

The Galaxy program uses a number of messages to inform the player of the current status of the game in progress, and to request information from the player about the move that is to be made next. These messages are stored in a large block of memory on pages 01 through 04. Each message is stored as a string of ASCII characters with a zero byte as the terminator for the message. There are a number of these messages that require the addition of variable information before the message is to be printed. These messages indicate the current status of the space ship which the player must keep watch over, the position of the objects in the galaxy, and the current progress of a specific move, such as the energy used or the tracking of a torpedo as it moves through a quadrant. The text of these messages is presented next with the location of the variable data indicated by X's.

“DO YOU WANT TO GO ON A SPACE VOYAGE?”

“YOU MUST DESTROY XX ALIEN SHIPS IN XX STARDATES
WITH X SPACE STATIONS”

“ - 1 - - 2 - - 3 - - 4 - - 5 - - 6 - - 7 - - 8 - ”

“X ” (Short Range Scan Row)

“STARDATE 30XX”

“CONDITION XXXXX” (Green or Red)

“QUADRANT X,X”

“SECTOR X,X”

“ENERGY XXXX”

“TORPEDOES XX”

“SHIELDS XXXX”

“COMMAND?”

“COURSE (1-8.5)?”

“WARP FACTOR (0.1-7.7)?”

“L.R. SCAN FOR”

“1 XXX 1 XXX 1 XXX 1” (Long Range Scan Row)

“1 XXX 1 XXX 1 XXX 1 XXX 1 XXX 1 XXX 1 XXX 1”
(Galaxy Display Row)

“MISSION FAILED, YOU HAVE RUN OUT OF STARDATES”

“KA-BOOM, YOU CRASHED INTO A STAR.
YOUR SHIP IS DESTROYED.”

“YOU MOVED OUT OF THE GALAXY.
YOUR SHIP IS LOST. . .LOST”

“ABANDON SHIP! NO ENERGY LEFT!”

“CONGRATULATIONS, YOU HAVE ELIMINATED ALL OF
THE ALIEN SHIPS”

“LOSS OF ENERGY XXXX”

“DANGER - SHIELD ENERGY 000”

“SHIELD ENERGY TRANSFER = ”

“NOT ENOUGH ENERGY”

“TORPEDO TRAJECTORY: ”

“ALIEN SHIP DESTROYED”

“YOU MISSED! ALIEN SHIP RETALIATES”

“SPACE STATION DESTROYED”

“TRACKING: X,X”

“GALAXY DISPLAY”

“PHASOR ENERGY TO FIRE = ”

“ALIEN SHIP AT SECTOR X,X:”

“ENERGY = XXXX”

“NO ALIEN SHIPS! WASTED SHOT”

“NO TORPEDOES”

“LAST SPACE STATION DESTROYED”

“CHICKEN!”

These messages require 1K bytes of memory to store one byte at a time. The text of many of these messages can be changed by the

reader to indicate varying degrees of emotion if desired. Or, if the user provided I/O routines require more than the amount of memory allocated, several of the messages can be shortened, or, if necessary, deleted, to make room for the I/O programming. If the messages are changed, the addresses in the program that refer to them must also be changed. These locations in the program will be indicated when the operating portion of the program is presented.

SUBROUTINES FOR THE GALAXY PROGRAM

There are many subroutines in this program. They are written to perform various tasks common to many of the routines throughout the Galaxy program. Among the types of functions they perform are outputting messages to the printer device, converting binary numbers to decimal (and vice-versa), setting up message contents with data to be displayed, controlling the movement of objects in the galaxy, and controlling the transfer of energy within the space ship. The subroutines of the Galaxy program reside in 1¼K bytes of memory on pages 05 through 09. This is equal to the amount of memory the operating portion of the program requires. Thus, one can see that the Galaxy program relies heavily on the subroutines to allow it to fit within 4K of memory. This section provides the details on the purpose and operation of the subroutines used in the Galaxy program.

The majority of the messages in the Galaxy program are outlined by means of the subroutine labeled MSG. This routine, presented below, outputs a string of ASCII characters stored in memory to the output device. MSG begins its output with the character pointed to by the index register when it is called. It will continue to output characters by calling the PRINT routine until a zero byte is encountered in the character string. The routine then returns to the calling program.

MSG	LDAA X	Fetch indexed character
	BEQ MSG1	Character = zero byte? Return
	JSR PRINT	No, output character
	INX	Advance to next character
	BRA MSG	Continue printout
MSG1	RTS	Output complete, return

The next subroutine is a random number generator used to provide random locations for the initial galaxy setup. It is also used in the placement of the alien ships, stars, and space stations each time a quadrant is entered by the space ship. The amount of energy an alien ship contains is also set up by calling on the random number subroutine. This random number routine provides a variation of numbers sufficient for use in the Galaxy program, and it can be applied to other programs requiring random number selection. The listing for the RN routine is presented next.

RN	LDAA RNM	Random number subroutine
	ROLA	Fetch random number and
	EORA RNM	Perform a series of
	RORA	Operations to generate
	INC RNM+\$1	A random value
	ADDA RNM+\$1	
	BVC SKIP	
	DEC RNM +\$1	
SKIP	STAA RNM	Store new random number
	RTS	Return with random number in A

The Galaxy program performs a number of operations involving the conversion of numbers from binary to decimal and vice versa for inputting and outputting numbers. The next trio of subroutines performs the conversion of double precision binary whole numbers to and from decimal, and also checks that digits entered on the keyboard fall within the range of the ASCII code for digits, namely B0 through B9. The binary-to-decimal routine, labeled BINDEC, converts a single or double precision binary number to its decimal equivalent up to five digits long, and stores the result in locations 51 through 55 on page 00. Accumulator B is set to 01 for a single precision number, and 02 for a double precision number, and the index register is set to the least significant byte of the number to be converted before the BINDEC subroutine is called. The decimal-to-binary subroutine, labeled DCBN, converts the decimal values stored in locations 51 through 54 on page 00 to the equivalent double precision binary number which is saved in location 28 for the least significant half, and 29 on page 00 for the most significant half of the binary value. The FNUM subroutine checks the memory location indicated by the index register for a valid ASCII digit, and returns with the N flag reset if it is a valid digit, or set if it is not. The listing for these subroutines is presented next.

BINDEC	STX PNTR1	Save pointer temporarily
	LDX PDG1ST	Set pointer to start of decimal table
	CLR X	Clear digit table
	CLR \$01,X	
	CLR \$02,X	
	CLR \$03,X	
	CLR \$04,X	
	LDX PNTR1	Set pointer to binary value
	LDAA X	Get least significant half
	DECB	Single precision?
BNDC	BEQ BNDC	Yes, most significant half = 0
	LDAB \$01,X	No, get most significant half
	STAA STORE1	Store LS half in temporary storage
	STAB STORE1+\$1	Store MS half in temporary storage+1
	LDX # \$1027	Set up value for 10K
	STX STORE2	Store for subtract routine
	BSR BD	Calculate 5th digit
	STAB DGT5TH	Store value of 5th digit
	LDX # \$E803	Binary value for 1K
	STX STORE2	Store for subtract routine
BD	BSR BD	Calculate 4th digit
	STAB DGT4TH	Store value of 4th digit
	LDX # \$6400	Binary value for 100
	STX STORE2	Store for subtract routine
	BSR BD	Calculate 3rd digit
	STAB DGT3RD	Store value of 3rd digit
	LDAA # \$0A	LS half value of 10 decimal
	STAA STORE2	Store for subtract routine
	BSR BD	Calculate 2nd digit
	STAB DGT2ND	Store value of 2nd digit
BD1	LDAA STORE1	Get unit value
	STAA DGT1ST	Store value of 1st digit
	RTS	Return to calling program
	CLRBD	Clear decimal digit counter
	INCB	Increment decimal digit
	LDAA STORE1	Fetch the least significant half
	SUBA STORE2	Subtract LS half of constant
	STAA STORE1	Save LS half of result
	LDAA STORE1+\$1	Fetch most significant half
	SBCA STORE2+\$1	Subtract MS half of constant
	STAA STORE1+\$1	Save MS half of result
	BCC BD1	If greater than 0, continue subtraction
	LDAA STORE1	Else, restore binary value
	ADDA STORE2	Add LS half back to result
	STAA STORE1	Restore result in memory
	LDAA STORE1+\$1	Fetch MS half of result
	ADCA STORE2+\$1	Add MS half back to result
	STAA STORE1+\$1	Restore result in memory
	DECB	Decrement decimal digit to correct
	RTS	Return

DCBN	CLR STORE2+\$1	Clear MS half of result
	LDAA DGT1ST	Fetch units digit
	STAA STORE2	Store in work area
	LDAB DGT2ND	Fetch ten's digit
	BEQ DC1	Digit = 0? Yes, do 100's digit
	LDX #\$0A00	Binary value of 10
	STX STORE1	To be added B times
	BSR TOBN	Add 10's digit
DC1	LDAB DGT3RD	Get 3rd digit
	BEQ DC2	Digit = 0? Yes, do 1000's digit
	LDX #\$6400	Binary value of 100
	STX STORE1	To be added B times
	BSR TOBN	Add 100's digit
DC2	LDAB DGT4TH	Get 4th digit
	BEQ DC3	Digit = 0? Yes, finished
	LDX #\$E803	Binary value of 1000
	STX STORE1	To be added B times
	BSR TOBN	Add 1000's digit
DC3	RTS	Return, binary value in STORE1
TOBN	LDX #STORE2	Set pointer to binary value
	JSR TO1	Add value to STORE1
	DECB	Multiplier = 0?
	BNE TOBN	No, continue
	RTS	Yes, return
FNUM	LDAA X	Fetch ASCII character
	CMPA #\$B0	Is character a number?
	BMI FNUM1	No, return with N flag set
	SUBA #\$BA	Valid number, return with
	ADDA #\$80	N flag reset
FNUM1	RTS	

Setting up the sector location of the stars, alien ships, and space station within a quadrant each time the space ship enters a new quadrant is performed by use of the following group of subroutines. When a game is started, the galaxy contents are set up in the last 64 bytes of page 00. The initial quadrant location of the space ship is then set, and the quadrant contents are moved from the galaxy content table to location 35 on page 00 by the QCNT subroutine. The NWQD subroutine is then called to set the location of the stars, space station, and alien ships in the quadrant. NWQD begins by clearing the sector locations of the galactic objects by storing C0 in locations 37 through 41 on page 00. It then determines how many of each object is contained in the quadrant, and calls on LOCSET to set the exact sector location of each. As each location is set, it is checked against the locations of the other objects in the quadrant by the MATCH

subroutine. If the new location is already assigned to another object, LOCSET selects a new location. As the final step in the NWQD subroutine, the energy in the shields of the alien ships is set to random levels from 0 to 1023 in the data table. After the game is underway, these same subroutines are called to set up the quadrant each time a new quadrant is entered. The LOAD subroutine is called at the start of the game and each time the space ship docks with the space station to restore its energy and set the torpedo count to ten. The listings of these subroutines are presented next.

NWQD	LDX PSOLSS	Set pointer to star table
	LDAA #\$C0	Clear code in A
	LDAA #\$0B	Counter in B
CLR1	STAA X	Clear object location table
	INX	Increment table pointer
	DECB	Decrement counter
	BNE CLR1	Not done? Clear more
	LDAB CQC	Else get quadrant contents
	ANDB #\$07	Get number of stars
	BEQ NWQD1	If none, check space station
	LDX PSOLSS	Set pointer to star table
	BSR LOCSET	Set up star locations
NWQD1	LDAA CQC	Get quadrant contents
	JSR ROTR3	Move to space station position
	TAB	Set up for LOCSET
	ANDB #\$01	Any space stations?
	BEQ NWQD2	No, check alien ships
	LDX PSLSS	Fetch space station table location
	BSR LOCSET	Set position if present
NWQD2	LDAA CQC	Get quadrant contents
	JSR ROTR4	Position alien count
	TAB	Put count in B
	ANDB #\$03	Mask for count
	BEQ LLAS	No aliens, skip positioning
	LDX PSLAS1	Set pointer to alien ship location
	BSR LOCSET	Assign alien ship locations
LDAS	BSR LLAS	Get random numbers
	LDX PVASE1	Pointer to alien ship no. 1 shields
	BSR LAS	Store alien ship no. 1 energy
	LDX PVASE2	Pointer to alien ship no. 2 shields
	BSR LAS	Store alien ship no. 2 energy
	LDX PVASE3	Pointer to alien ship no. 3 shields
LAS	STAA X	Store least significant half value

	ANDA # \$03	Mask for most significant half
	STAA \$01,X	Store most significant half
LLAS	JMP RN	Get random and return
LOCSET	STX PNTR1	Store table pointer
	BSR LLAS	Fetch random location
	ANDA # \$3F	Mask off most significant bits
	BSR MATCH	New location match others?
	BEQ LOCSET+\$2	Yes, find new location
	LDX PNTR1	Fetch table pointer
	STAA X	Store random location
	INX	Table pointer to next object
	DECB	Decrement object counter
	BNE LOCSET	Counter not =0, do next
	RTS	Table complete, return
MATCH	LDX PSOLSS	Set pointer to star table
MATCH2	CMPA X	Same sector location?
	BEQ MATCH1	Yes, match, return
	INX	Advance table pointer
	CPX PDVME	End of table?
	BNE MATCH2	No, check next
	INX	Yes, reset Z flag
	DEX	
MATCH1	RTS	Return
QCNT	LDAA CQLSS	Fetch current quadrant
	ORAA # \$C0	Form pointer to galaxy
	JSR ATINX1	Set pointer to quadrant
	LDAA X	Fetch quadrant contents
	STAA CQC	Store new quadrant contents
	RTS	Return
LOAD	LDX # \$8813	Store double precision value 5000
	STX DVME	In main energy store
	LDX # \$0000	Set shields to zero
	STX DVSE	
	LDAA # \$0A	Load ten torpedoes on board
	STAA NTR	
	RTS	Return to main program
ATINX1	CLR PNTR1	Clear M.S. half of pointer for page 00
ATINX	STAA PNTR1+\$1	Store A in least significant half of pntr
	LDX PNTR1	Load pointer into index register
	RTS	

The next group of subroutines is called to indicate to the operator that the game has ended due to the occurrence of one of the follow-

ing problems. Either the stardate time has run out (TIME), or the space ship has moved out of the known galaxy (LOST), or the space ship has smashed into a star (WPOUT), or the space ship has run out of energy (EOUT). These subroutines print an advisory message, and then jump to the beginning of the program to inquire whether the operator desires to try again. The listings for these subroutines are presented next.

TIME	LDX #\$025D	Stardates time run out - player loses
DONE	JSR MSG JMP GALAXY	Print message and start A new game
LOST	LDX #\$02C8 BRA DONE	Out of known galaxy Player loses
WPOUT	LDX #\$028D BRA DONE	Smashed into star Player loses
EOUT	LDX #\$0497 BRA DONE	Out of energy Abandon ship

The next group of subroutines deals with setting up various messages for the output display device. The first subroutine, DIGPRT, fetches a digit stored in memory, forms the ASCII equivalent, and stores the ASCII code in the message to be printed. The index register must be set to the location of the least significant digit in the message, and the accumulator B must contain a binary count of the number of digits to be placed in the message. The BCD digits must reside in locations 51 through 54 on page 00 before this subroutine is called. The listing for this subroutine is now presented.

DIGPRT	STX PNTR1	Save message digit location pointer
	STS PNTR2	Save stack pointer temporarily
	LDS PNTR1	Set stack pointer to 1st digit location
	LDX PDG1ST	Set index to 1st digit
DGPRT1	LDAA X	Get digit from storage
	INX	Advance index to next digit
	ORAA #\$B0	Form ASCII code
	PSHA	Store in message
	DECB	Decrement digit counter
	BNE DGPRT1	Not =0? Continue
	LDS PNTR2	Equals 0, restore stack pointer
	RTS	And return

ROWSET is used by the short range scan routine to set up the contents of each row before it is printed. ROWSET first clears the row message by filling it with space characters. It then stores the ASCII code for the row number at the beginning of the message. The location of all of the objects contained in the quadrant is then checked to determine whether they are present in the row being prepared for output. If one or more of the objects are located in the row, the ASCII code for the symbolic representation of each is stored in the row message at the proper column location. The subroutine RWPNT is used to check for the location of each object, and to set a pointer to the column location within the row message for storage of the object's ASCII representation. When ROWSET is called, the B accumulator must contain the binary value of the row number. When the row message is set up, the MSG subroutine is called to print it. The listings for these two subroutines are given next.

ROWSET	LDX #\$018F	Set pointer to row message
	LDAA #\$A0	Clear with ASCII space
RCLR	STAA X	Store space character
	INX	Advance pointer
	CPX #\$01A7	Message cleared?
	BNE RCLR	No, clear next
	TBA	
	ORAA #\$B0	Form ASCII code for row
	STAA #\$018E	Store in message
	DECB	Set up row number for checkout
	LDX PSLOSS	Set pointer to object location table
	BSR RWPNT	Fetch space ship location
	BNE STR	In this row?
	LDAA #\$BC	Yes, store space ship
	STAA X	Code for printout
	LDAA #\$AA	
	STAA \$01,X	
	LDAA #\$BE	
	STAA \$02,X	
STR	LDX PSOLSS	Set a star table pointer
	STX PNTR2	
STR1	BSR RWPNT	Is star in this row?
	BNE NXSTR	No, pointer to next star
	LDAA #\$AA	Yes, store star code
	STAA \$01,X	In proper location
NXSTR	INC PNTR2+\$1	Increment star table pointer
	LDX PNTR2	Put new pointer in index

	CPX PSLSS	End of star table?
	BNE STR1	No, check next star
	BSR RWPNT	Space station in this row?
	BNE AS	No, look for alien ships
	LDAA # \$BE	Yes, store space station
	STAA X	Code for printout
	LDAA # \$B1	
	STAA \$01,X	
	LDAA # \$BC	
	STAA \$02,X	
AS	LDX PSLAS1	Set alien ship table pointer
	STX PNTR2	
AS1	BSR RWPNT	Alien ship in this row?
	BNE NXAS	No, look for next ship
	LDAA # \$AB	Yes, store code for alien
	STAA X	Ship printout
	STAA \$01,X	
	STAA \$02,X	
NXAS	INC PNTR2+\$1	Advance alien ship table pointer
	LDX PNTR2	Get new pointer
	CPX PDVME	End of table?
	BNE AS1	No, try next alien
	LDX # \$018C	Set pointer to print short range scan &
	JMP MSG	Return
RWPNT	LDAA X	Fetch entry location
	BMI RWPNT1	No, return
	JSR ROTR3	Position row entry
	ANDA # \$07	Separate row entry
	CBA	Is row = current row?
	BNE RWPNT1	No, return
	LDAA X	Yes, fetch column location
	ANDA # \$07	Separate column location
	STAA STORE1	Save column
	ASLA	Multiply by two
	ADDA STORE1	Form pointer to row message
	ADDA # \$8F	
	CLR PNTR1	Set up pointer storage to page 01
	INC PNTR1	
	JSR ATINX	Set index pointer from A
	CLRA	Set zero flag
RWPNT1	RTS	Return
ROTR4	ASRA	Shift accumulator A right
ROTR3	ASRA	
	ASRA	
	ASRA	
	RTS	Return

The subroutine labeled QUAD is used to place the row and column location of the current quadrant into the QUADRANT R,C message. The quadrant message is used in the short range scan and in the heading for the long range scan. It fetches the quadrant location from the data table and stores the ASCII code for the row and column numbers in the message. It then calls MSG to print it. The subroutine TWO is called to separate the row and column numbers and store them in the proper locations in the message. TWO is also used to place the row and column location of the current sector in the SECTOR R,C message, which is part of the short range scan routine. The listings for QUAD and TWO are presented next.

QUAD	LDX #\$01D4	Store temp, quadrant
	STX PNTR1	Message pointer
	LDX #CQLSS	Index to quadrant location storage
	BSR TWO	Put digits in message
	LDX #\$01C9	Index to quadrant message
	JMP MSG	Print quadrant message and return
TWO	LDAA X	Fetch row and column
	TAB	Save row and column
	LDX PNTR1	Get message pointer
T1	JSR ROTR3	Position row number
	ANDA #\$07	Mask off other bits
	ADDA #\$B1	Form ASCII code
	STAA X	Store in message
	ANDB #\$07	Separate column number
	ADDB #\$B1	Form ASCII code
	STAB \$02,X	Store column in message
	RTS	

The final three subroutines of this group are used in the preparation and output of the long range scan and the galaxy display. The NTN subroutine prints the dividing line between rows for each of the printouts mentioned. It first outputs a carriage return/line feed combination, and then prints as many hyphens as are defined in accumulator B. QDSET takes the quadrant contents stored in accumulator A and forms the ASCII code for the digits indicating the number of alien ships, space stations, and stars in the quadrant, and stores them in the message indicated by the index register. QDSET is called by the galaxy display routine and LRR. LRR is a subroutine for the long range scan routine that sets up each row of the scan for printout. The quadrant location in the center of the long range row being prepared is contained in the A accumulator when LRR is called. If

the left hand quadrant is outside the galaxy, it is set up to be printed as all zeros. When the long range row is completed, the MSG routine is called to output the row to the display device. The listings for these subroutines are presented next.

NTN	LDAB #\$13	Set counter 19 dashes
NT1	LDAA #\$8D	Print carriage return
	BSR NT3	
	LDAA #\$8A	Print line feed
	BSR NT3	
NT2	LDAA #\$AD	ASCII code for dash
	BSR NT3	Print '.'
	DECB	Decrement counter. =0?
	BNE NT2	No, print more dashes
	RTS	Yes, return
NT3	JMP PRINT	
QDSET	TAB	Fetch quadrant contents
	JSR ROTR4	Position alien ship number
	ANDA #\$03	Mask alien ship number
	ORAA #\$B0	Form ASCII digit
	STAA X	Store in message
	TBA	Fetch quadrant contents
	JSR ROTR3	Position space ship number
	ANDA #\$01	Mask space ship number
	ORAA #\$B0	Form ASCII digit
	STAA \$01,X	Store space ship in message
	ANDB #\$07	Mask star number
	ORAB #\$B0	Form ASCII digit
	STAB \$02,X	Store in message
	RTS	Return
CLC1	CLRA	Clear column contents
	BRA LR3	Print 000 quadrant
CLC2	CLRA	Clear column contents
	BRA LR4	Print 000 quadrant
LR5	JMP ATINX1	
LRR	ORAA #\$C0	Set pointer to galaxy
	TAB	
	STAA STORE1	Save pointer
	ANDB #\$07	First column?
	BEQ CLC1	Yes, first column zero
	DECA	No, back one column
	BSR LR5	Set quadrant pointer
	LDAA X	Fetch quadrant contents
LR3	LDX #\$04C9	Set pointer to left quadrant
	BSR QDSET	Set quadrant contents

	LDAA STORE1	Get pointer
	BSR LR5	Set quadrant pointer
	LDAA X	Fetch quadrant contents
	LDX #\$04CF	Set pointer to middle quadrant
	BSR QDSET	Set quadrant contents
	LDAA STORE1	Fetch quadrant location
	TAB	
	ANDB #\$07	Is quadrant in last column?
	CMPB #\$07	
	BEQ CLC2	Yes, right column =0
	INCA	Set to right quadrant
	BSR LR5	Set quadrant pointer
	LDAA X	Fetch quadrant contents
LR4	LDX #\$04D5	Index to right quadrant
	JSR QDSET	Set quadrant contents
LRP	LDX #\$04C5	Set pointer to long range row message
LR6	JMP MSG	Print and return

The depletion of energy from the space ship's main storage bank and its shields is an important function in this program. The following group of subroutines is called to delete the energy from the ship, and to check the energy level of the ship. The subroutine labeled ELOS deletes the amount of energy contained in the index register from the ship's protective shields. The least significant half of the energy to be deleted must be stored in the most significant half of the index register, and the most significant half of the energy must be stored in the least significant half of the index register. By switching the two eight bit values around in this manner, the STX instruction will store the energy in the desired order in memory (the least significant half in the lower address of the pair of memory locations). The amount of energy deleted is first output to the display device to inform the operator of the loss. The shield energy level is checked, and if sufficient, the energy is removed from the shield. If the level is not high enough to absorb the loss, the remaining shield energy is transferred to the main supply, and the loss is taken from the main storage bank. If at this time the main supply is not enough, the ship is out of energy, and the game is over. Otherwise, since the shield energy is zero, the warning message is output and an additional 25 percent of the energy loss is depleted from the main supply as a penalty. The listing of ELOS and its supporting subroutines is shown next.

ELOS	STX STORE3 LDX #STORE3 LDAB # \$02 JSR BINDEC LDX # \$0313 LDAB # \$04 JSR DIGPRT LDX # \$02FF BSR LR6 LDX STORE3 STX STORE1	Save energy value Set pointer to value to be converted Double precision conversion Convert to BCD for message Set pntr to least signif digit of energy Set digit counter Put digit in message Set pointer to energy loss message Print energy loss message Restore value for routines To follow
ELS1	BSR CKSD	Is shield energy sufficient?
	BCC FMSD	Yes, delete from shields and return
SDO1	LDX DVSE STX STORE1 BSR FMSD BSR TOMN LDX STORE3 STX STORE1	Move shield energy to main storage Remove energy from shields Move shield energy to main storage Fetch energy to be deleted Store for routines
SDO	BSR CKMN BCS EOUT1 BSR FMMN LDX # \$0315 BSR LR6 LDAB # \$02 BSR DVD BSR CKMN BCS EOUT1 BRA FMMN	Energy enough? No, ship out of energy Yes, take from main Print WARNING! DANGER - SHIELD ENERGY 000 Divide energy loss by 2 twice To divide by 4 Enough main energy for penalty? No, out of energy message Yes, take penalty and return
CKSD	LDX PDVSE BRA CK1	Check shield energy level Against requested level
CKMN	LDX PDVME	Check main energy level
CK1	LDAA \$01,X CMPA STORE1+\$1 BNE ELOS1	Fetch most significant half Is most significant half =0? No, return with flags set up
CK2	LDAA X CMPA STORE1 RTS	If >=, return with C flag set If less than, C flag reset Return
FMSD	LDX PDVSE BRA FM1	Set pointer to shield energy Subtract energy from shields
FMMN	LDX PDVME	Set pointer to main storage
FM1	LDAA X SUBA STORE1	Fetch least significant half of energy Subtract least significant half of loss

	STAA X	Return to storage
	LDAA \$01,X	Fetch most significant half of energy
	SBCA STORE1+\$1	Subtract most significant half of loss
	STAA \$01,X	Return to storage
	RTS	
TOSD	LDX PDVSE	Set pointer to shield energy
	BRA TO1	Add energy to shields
TOMN	LDX PDVME	Set pointer to main energy
TO1	LDAA X	Fetch least significant half of energy
	ADDA STORE1	Add least significant half of loss
	STAA X	Return to storage
	LDAA \$01,X	Fetch most significant half of energy
	ADCA STORE1+\$1	Add most significant half of loss
	STAA \$01,X	Return to storage
	RTS	
DVD	TSTB	Divide the double
	ROR STORE1+\$1	Precision value
	ROR STORE1	By two the number
	DECB	Of times indicated in B
	BNE DVD	
ELOS1	RTS	Return
EOUT1	JMP EOUT	

The removal of energy from the main supply for the execution of commands, firing phasors and torpedoes, and moving through the galaxy is provided by the ELOM subroutine. The amount of energy to be removed is stored in the index register (as described in the ELOS subroutine) when the ELOM subroutine is called. If the main energy bank contains enough energy, the energy is deleted, and the subroutine returns to the calling program. If there is not enough energy, the shield energy is transferred to the main storage bank in an effort to provide for the loss. If this does not provide sufficient energy, the game is over. However, if the transfer does produce the energy needed in the main supply, the energy will be removed; and since the shield energy has been reduced to zero, an additional 25 percent of the energy loss will be deleted from the main supply as a penalty. The listing for ELOM is presented next.

ELOM	BSR CKMN	Enough energy in main?
	BCC FMMN	Yes, take from main and return
	LDX STORE1	No, save value of energy loss
	STX STORE3	
	JMP SDO1	Transfer shield energy and try again

The amount of energy transferred to or from the shields and the energy to be fired by the phasor is entered by the operator. The EIN subroutine is called to input these energy values. The first entry is checked to determine whether it is a minus sign, used in the shield entry. Location 55 on page 00 will be all zeros if the value is to be positive, and non-zero for a negative entry. Each digit entered is checked for validity, and then the ASCII code is masked off, resulting in the BCD digits being stored in locations 54 through 51. The units digit is stored in location 51. Four digits must be entered by the operator when this routine is called. If the input is found to be invalid, the routine returns with the N flag set to '1.' If the input is valid, the N flag is reset upon returning to the calling program. The listing for this routine is presented next.

EIN	LDX PDG5TH	Set pointer to start of digit store
	CLR X	Clear sign indicator
	JSR INPUT	Get first character
	CMPA #\$AD	Negative sign?
	BNE EN2	No, check digit
	STAA X	Make negative indicator not =0
EN1	JSR INPUT	Get next character
EN2	DEX	Advance storage pointer
	STAA X	Store digit
	JSR FNUM	Valid digit?
	BMI EIN1	No, return with N flag set
	LDAA X	Yes, fetch digit
	ANDA #\$0F	Mask off ASCII bits
	STAA X	Save BCD value
	CPX PDG1ST	End of input?
	BNE EN1	No, fetch next digit
EIN1	RTS	Yes, return

When the space ship destroys an alien ship or space station, the result is the elimination of the alien ship or space station from the galaxy. The subroutine DLET is called to perform this function. First, the sector location of the object is cleared by storing a C0 in the data table at the location indicated by the index register. From this location, the identity of the object to be deleted is ascertained. A pointer is then formed indicating the location of the quadrant in the galaxy content table from which the object is to be removed. If the object was a space station, it is removed from the galaxy and the number of space stations is decremented. If this value goes to zero, a warning message is output to inform the operator that the last space

station has been destroyed. If an alien ship is destroyed, it is removed from the galaxy and its count is decremented. When the number of alien ships reaches zero, the game is over and the operator has successfully completed the mission. The listing of DLET is shown next.

DLET	LDAA #\$C0	Load with clear character
	STAA X	Clear object from table
	STX PNTR2	Save table location
	LDAA CQLSS	Get quadrant location
	ADDA #\$C0	Form galaxy table pointer
	JSR ATINX1	Place pointer in index
	STX PNTR3	Set table pointer
	LDX PNTR2	Fetch table location
	JSR COMPAR	Space station hit?
	BNE DLAS	No, delete alien ship
	LDX PNTR3	Fetch galaxy pointer
	LDAA X	Get quadrant contents
	ANDA #\$37	Delete space station
	STAA X	Restore quadrant in galaxy
	STAA CQC	Place new contents
	DEC NSS	Decrement number of space stations
	BNE DLET1	If more left, return
	LDX #\$04DB	If number of space stations =0,
	JMP MSG	Print warning message & return
DLAS	LDX PNTR3	Fetch galaxy pointer
	LDAA X	Get quadrant contents
	SUBA #\$10	Delete 1 alien ship from quadrant
	STAA X	Restore to galaxy
	STAA CQC	Save new contents
	DEC NAS	Decrement number of alien ships
	BNE DLET1	More aliens, return
	LDX #\$03D4	All aliens destroyed!
	JMP DONE	Print CONGRATULATIONS,
DLET1	RTS	Start new game
COMPAR	STX PNTR1	Store index value
	LDAA PNTR1+\$1	Fetch low portion of the address
	CMPA #\$3E	Set flags for address relative to SLSS
	RTS	Return with results

The final group of subroutines to be presented deals with the movement of the space ship through the galaxy, and the tracking of the torpedo within the quadrant. Moving an object through the galaxy is performed with the use of a table referred to as the COURSE TABLE. The course table, presented next, is located at the

beginning of page 00, and contains 16 pairs of row and column displacement values. There is one pair of displacement values for each possible direction of movement. The first value of each pair is the column displacement; the second value is the row displacement. The entries in the course table are made up of the binary values 2, 1, 0, -1, and -2. A displacement of 1 advances the object one half of a sector for each sector move made. So, for example, if the course was chosen as 8.5, the displacement value for the column is two, and for the row is one. This means that for every column moved to the right, the object would move one half of a row down. A move is made by the program by separating the row and column location of the object to be moved, rotating each to the left once, and using the adjusted values to calculate the move. Then, for each sector move made, the row and column displacement is added to the adjusted row and column location. When the move is completed, the adjusted values are rotated to the right once, and then combined to give a new sector location to the object. By using this method it is possible for the direction of travel to be broken down to every $22\frac{1}{2}$ degrees.

DISPLACEMENT VALUES

COURSE SELECTED

02	1.0
00	
02	1.5
FF	
02	2.0
FE	
01	2.5
FE	
00	3.0
FE	
FF	3.5
FE	
FE	4.0
FE	
FE	4.5
FF	
FE	5.0
00	
FE	5.5
01	

FE	6.0
02	
FF	6.5
02	
00	7.0
02	
01	7.5
02	
02	8.0
02	
02	8.5
01	

The subroutine DRCT is called to input the course direction from the operator through the input device. The two digits defining the move are checked for validity when entered, and then used to form a pointer to the course table. If the input is valid, the routine returns with the Z flag reset, and the pointer stored in location 20 on page 00. If invalid, the Z flag is set before returning. The ACTV subroutine is then called to fetch the displacement values from the course table, and store the column displacement in location 2D and the row displacement in location 2C. It then sets up the adjusted row and column values, and stores them in locations 2E and 2F respectively.

The subroutine labeled TRK is called to make the individual sector moves. First, the quadrant crossing flag is cleared. The column displacement is then added to the adjusted column location, and a quadrant crossing to the left or right is checked. If the crossing did occur, the crossing flag is set, and the adjusted column is corrected to indicate the new column value. The crossing is then checked for a move out of the galaxy, which would be indicated by the TRK subroutine returning with the Z flag set. If the move is not out of the galaxy, the new quadrant location is stored at location 4C on page 00. The row displacement is then added to the adjusted row location, and a quadrant crossing up and down is checked. If a quadrant is crossed, the crossing flag is set and a move out of the galaxy is checked. If the crossing is out of the galaxy, the routine returns with the Z flag set. Otherwise, the new quadrant location is stored at location 4C, and the routine returns with the Z flag reset. The final subroutine of this group is called RWCM, and is called to restore the adjusted row and column locations to the single byte used to define the

final location of the object moved. The listings for these subroutines are presented next.

DRCT	JSR INPUT CMPA #\$B1 BCS ZRET CMPA #\$B9 BCC ZRET ANDA #\$0F ASLA TAB LDAA #\$AE JSR PRINT JSR INPUT CMPA #\$B0 BEQ CR1 CMPA #\$B5 BNE ZRET	Input first course number Is input less than 1? Yes, illegal input Is input greater than 8? Yes, illegal input No, mask off ASCII bits If good times 2 And save in temporary storage Print decimal point Input second course number Is digit zero? Yes, continue process No, is digit =5? No, return with Z flag set
CR1	ANDA #\$01 ABA ASLA SUBA #\$04 STAA PNTR1+\$1 CLRA STAA PNTR1 INCA RTS	Mask off all but first bit Add first number input And form pointer to course table Save pointer in temporary storage Clear least significant byte of pointer Reset Z flag Before returning
ZRET	CLRA RTS	Set Z flag And return
ACTV	STS PNTR2 LDS PST51 LDAA SLOSS TAB ANDB #\$07 ASLB PSHB ANDA #\$38 LSRA LSRA PSHA LDX PNTR1 LDAA X PSHA LDAA \$01,X PSHA	Save stack pointer temporarily Set stack to storage area Get present location Save temporarily Mask out column Multiply by 2 Store adjusted column Mask out row Set up times 2 value Save adjusted row Get displacement table pointer Get column movement Store column displacement Get row movement Store row displacement

	LDS PNTR2 RTS	Restore stack pointer
TRK	CLR CF LDAA STORE5+\$1 ADDA STORE4+\$1 STAA STORE5+\$1 BPL NOBK ANDA #\$0F STAA STORE5+\$1 INC CF LDAA CQLSS ANDA #\$07 BEQ TRK1 DEC CQLSS BRA RMV	Clear quadrant crossing flag Get adjusted column Add column move Save temporarily current column If no left crossing, branch Left crossing correction And save new adjusted column Indicate left crossing And decrement current column Is CQC = 0? Yes, return with Z set No, decrement CQC Do row move
NOBK	CMPA #\$10 BCS RMV ANDA #\$0F STAA STORE5+\$1 INC CF LDAA CQLSS ANDA #\$07 INCA CMPA #\$08 BEQ TRK1 INC CQLSS	Quadrant crossing right No, do row move Yes, correct and Save new adjusted column Indicate crossing by making Crossing flag non-zero Fetch current quadrant location Separate column entry Increment column entry Move out of galaxy Yes, return with flags set No, increment quadrant column
RMV	LDAA STORE5 ADDA STORE4 STAA STORE5 BPL NOUP ANDA #\$0F STAA STORE5 INC CF LDAA CQLSS TAB ANDA #\$38 BEQ TRK1 SUBB #\$08 STAB CQLSS BRA CKX	Get adjusted row value Add movement Save new adjusted row If not up, jump Move up one quadrant, correct And save new adjusted value Make crossing flag non-zero Decrement quadrant row Save temporarily Is quadrant row = 0? Yes, return with Z flag set No, decrement current quadrant row Save new current quadrant Then perform crossing logic
NOUP	CMPA #\$10 BCS CKX ANDA #\$0F STAA STORE5 INC CF	Quadrant crossing down? No, check for crossing flag Yes, correct and Save new adjusted row Indicate crossing

	LDAA CQLSS	Then increment quadrant row
	TAB	Save temporarily
	ANDA # \$38	Separate row entry
	ADDA # \$08	Increment row value
	CMPA # \$40	Out of galaxy?
	BEQ TRK1	Yes, return with Z flag set
	ADDB # \$08	No, increment row
	STAB CQLSS	Save new current quadrant
CKX	BNE TRK1	Return with Z flag reset
	LDAA # \$01	If not, reset it
TRK1	RTS	
RWCM	LDAA STORE5+\$1	Fetch adjusted column
	LSRA	Adjust position
	ANDA # \$07	Form column value
	LDAB STORE5	Fetch row
	ASLB	Position row value
	ASLB	
	ANDB # \$38	Form row value
	ABA	Form row and column byte
	RTS	Return

MAJOR ROUTINES OF THE GALAXY PROGRAM

The main portion of the Galaxy program consists of nine major functional routines. The first of these routines provides the initial galaxy setup. The contents of each quadrant are randomly selected from the galaxy setup table which consists of many possible quadrant content arrangements. The selected quadrant arrangements are then stored in the galaxy content table. This random selection provides a different game for the operator each time GALAXY is played.

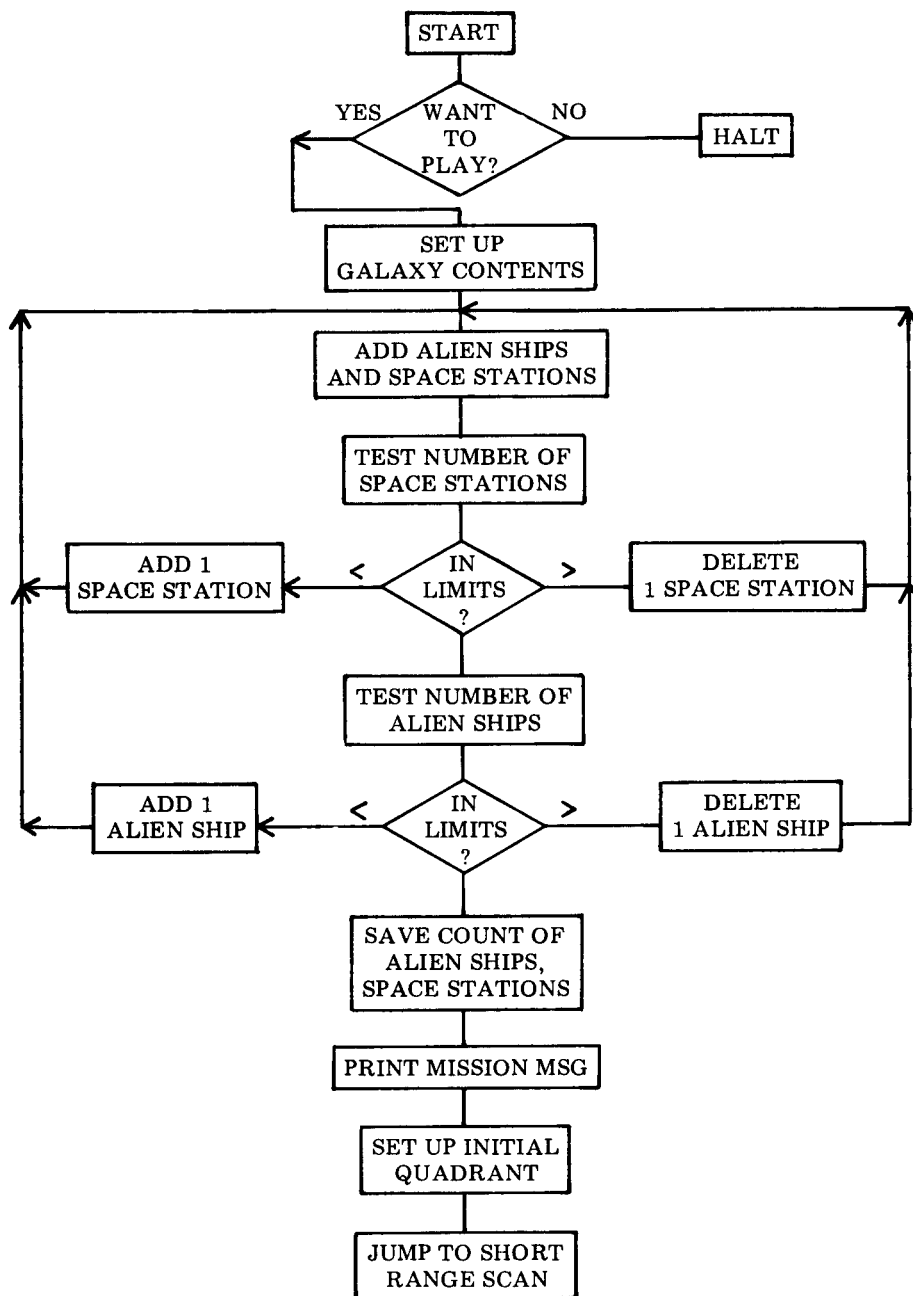
This routine, labeled GALAXY, is the starting point of the entire program. It begins by posing the question, "DO YOU WANT TO GO ON A SPACE VOYAGE?" The INPUT routine is then called to input the response from the operator. If the response is "N," the program outputs the message "CHICKEN," and jumps to an address set up by the programmer. To insert this jump, the user replaces the three NOP instructions with a jump instruction. For any response other than N, the program will store the code received in the second byte of the random number and proceed to form the galaxy contents to be used for this game.

With the use of the random number generator, various locations in the galaxy setup table are selected and stored in the galaxy content table. When the galaxy content table is filled, the number of alien ships and space stations in the newly formed galaxy is calculated. If the count is not within the limits desired, the contents of the galaxy are revised until the proper limits are met. The number of alien ships must be between 10 and 31, and the number of space stations must be between 2 and 6. These limits may be revised by the reader by simply changing the binary values in the compare instructions which set the limits. Once the galaxy is completed, the values indicating the number of space stations and alien ships are stored in the data table. The number of stardates is then set to a value of five greater than the number of alien ships, and is also stored in the data table. The message stating the mission assigned for this game is then prepared by storing the ASCII code for the number of alien ships, stardates, and space stations in the body of the message, and calling the MSG subroutine to output it. This routine finishes by selecting the starting quadrant, loading the initial energy and torpedoes for the space ship, setting up the locations of the quadrant contents, and

setting the start location of the space ship within the quadrant. The flow chart and program listing of this routine are presented next.

GALAXY	LDS #\$0EFF	Set stack pointer to stack area
	JMP START	Jump to start of Galaxy program
START	LDX #\$0100	Set pointer to initial message
	JSR MSG	Print introduction
	JSR RN	Increment random number
	JSR INPUT	Input character
	STAA RNM+\$1	Store input to randomize
	CMPA #\$CE	Character = N? Yes, stop game
	BNE OVER	No, set up galaxy
	LDX #\$04E2	Print "CHICKEN"
	JSR MSG	
	NOP	User defined
OVER	NOP	End of program
	NOP	
OVER	LDAB #\$C0	Set pointer to galaxy storage
	STAB STORE1	Save in temporary storage
GLXSET	JSR RN	Fetch random number
	ANDA #\$7F	Form pointer to
	LDAB #\$0F	Galaxy table from
	STAB PNTR1	Random number
	JSR ATINX	Set index to galaxy table
	LDAB X	Get galaxy entry
	LDAA STORE1	
	JSR ATINX1	Set index to galaxy content table
	STAB X	Store quadrant contents
	INC STORE1	Galaxy contents complete?
GLXSET	BNE GLXSET	No, fetch more sectors
GLXCK	CLR NSS	Clear space station count
	CLR NAS	Clear alien ship count
	LDX #\$00C0	Pointer to galaxy content table
GLXCK1	LDAA X	Fetch quadrant contents
	TAB	Save in 'B' accumulator
	ANDA #\$08	Mask space station
	ADDA NSS	Add to space station total
	STAA NSS	Save space station total
	ANDB #\$30	Mask alien ship
	LSRB	Position
	LSRB	
	ADDB NAS	Add to alien ship total

	STAB NAS	Save alien ship total
	INX	Increment galaxy content pointer
	CPX #0100	End of table?
	BNE GLXCK1	No, continue adding
	LDAA NSS	Fetch space station total
	LSRA	Position total to right
	LSRA	
	LSRA	
	STAA NSS	Store total
	CMPA #07	Too many space stations?
	BPL SSPLS	Yes, delete 1
	CMPA #02	Too few?
	BPL CAS	No, O.K., check alien ships
SSMNS	LDAB #08	Yes, form mask to
	STAB STORE1	Add one space station
	BRA MNS	
SSPLS	LDAB #F7	Form and store mask to
	STAB STORE1	Delete one space station
	BRA PLS	
ASPLS	LDAB #CF	Form mask to delete
	STAB STORE1	One alien ship
PLS	JSR RN	Fetch random number
	ORAA #C0	Form galaxy table pointer
	JSR ATINX1	Place pointer in index
	LDAA STORE1	Fetch mask
	ANDA X	Delete from galaxy
PLS1	STAA X	Store new quadrant contents
	JMP GLXCK	Check galaxy again
ASMNS	LDAB #10	Form mask to add
	STAB STORE1	One alien ship
MNS	JSR RN	Fetch random number
	ORAA #C0	Form galaxy table pointer
	JSR ATINX1	Place pointer in index
	LDAA STORE1	Fetch mask
	ORAA X	Add one alien ship to quadrant
	BRA PLS1	Check galaxy again
CAS	LDAA NAS	Fetch alien ship total
	LSRA	Position
	LSRA	
	STAA NAS	Save total
	CMPA #20	Too many alien ships?
	BPL ASPLS	Yes, delete one alien ship
	CMPA #0A	Too few?
	BMI ASMNS	Yes, add one alien ship

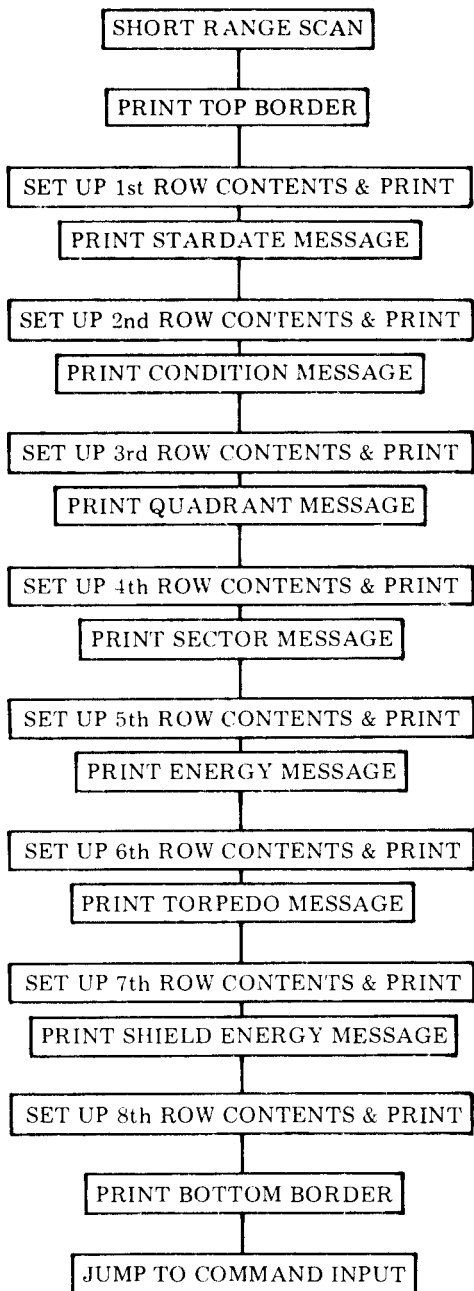


LDAA #\$05	Set up five more stardates
ADDA NAS	Than alien ships
STAA NSR	Save number of stardates
LDX #NSR	Convert binary value
LDAB #\$01	Set precision counter
JSR BINDEC	Convert stardate value
LDX #\$014E	Pointer to stardate count
LDAB #\$02	Set precision counter
JSR DIGPRT	Put digits in starting message
LDX #NAS	Pointer to alien ship value
LDAB #\$01	Set precision counter
JSR BINDEC	Convert alien ship value
LDX #\$013C	Pointer to alien ship count
LDAB #\$02	Set precision counter
JSR DIGPRT	Put digits in starting message
LDAA NSS	Get number of space stations
ORAA #\$B0	Form ASCII digit
STAA \$015F	Store in starting message
LDX #\$0128	Pointer to start of message
JSR MSG	Print starting message
JSR RN	Fetch starting quadrant
ANDA #\$3F	Mask off MSB's
STAA CQLSS	Save current quadrant location
JSR QCNT	Fetch current quadrant contents
JSR LOAD	Set initial conditions
JSR NWQD	Set quadrant contents location
LDX PSLOSS	Pointer to sector location storage
LDAB #\$01	Set precision counter
JSR LOCSET	Set initial space ship location

The next routine, which immediately follows the galaxy setup routine, is the short range scan. The location of each of the objects contained in the current quadrant is displayed as illustrated in the sample short range scan in Chapter One. By the use of the ROWSET, BINDEC, DIGPRT, and MSG subroutines, each line of the scan is prepared and output to the display device. This routine is entered following the galaxy setup to display the initial quadrant; then after each move by the space ship either within the quadrant or when a new quadrant is entered, and in response to a command to display a short range scan. The flow chart and listing for this routine, which begins at the label SRSCN, are presented next.

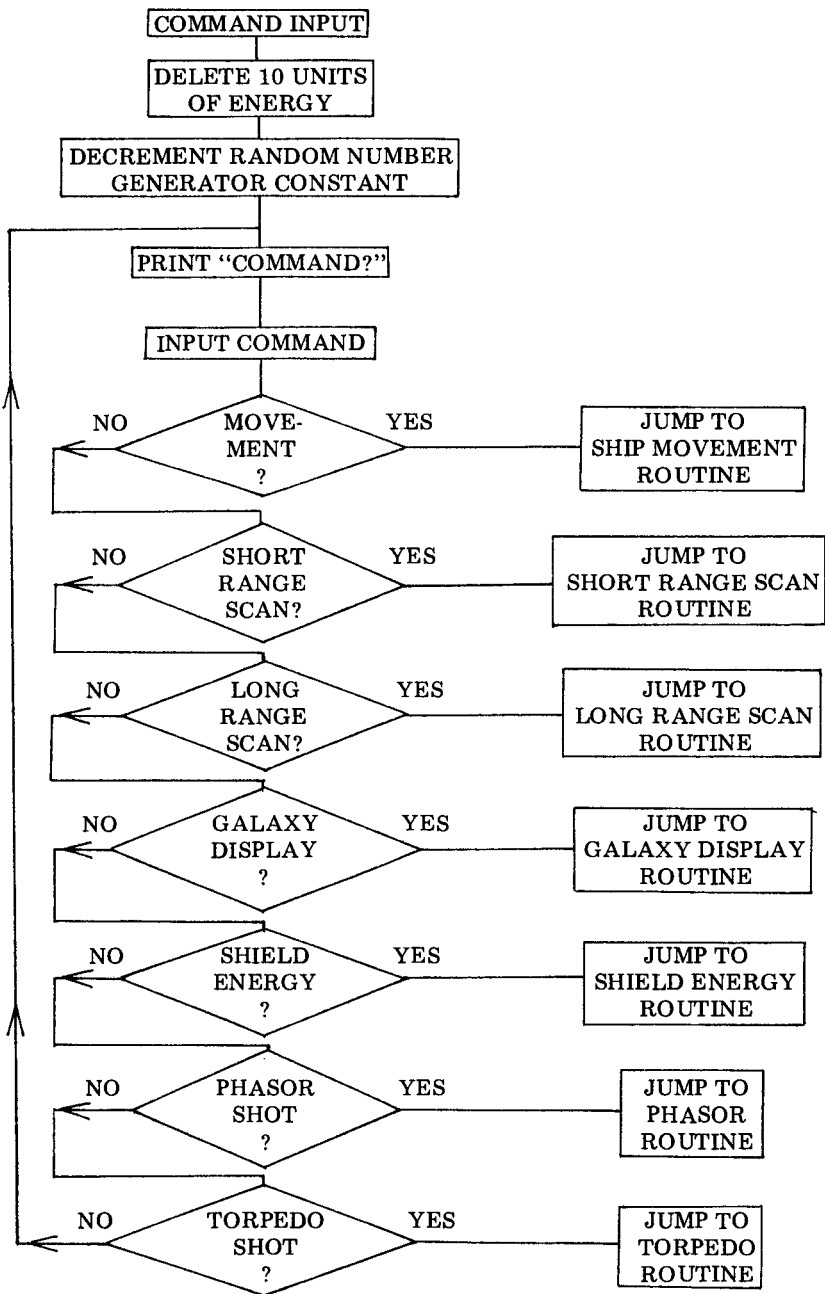
SRSCN	LDX #\$0170	Set pointer for short range scan
	JSR MSG	Print initial row
	LDAB #\$01	Set row number one
	JSR ROWSET	Set up row for printout
	LDAA #\$32	
	SUBA NSR	Calculate stardate number
	STAA STORE1	Save temporarily

	LDX PSTR1	Set pointer to binary value
	LDAB # \$01	Set precision counter
	JSR BINDEC	Convert to current stardate
	LDX # \$01B6	Set pointer to stardate message
	LDAB # \$02	Set counter to number of digits
	JSR DIGPRT	Put digits in stardate message
	LDX # \$01A8	Set pointer to message
	BSR SRSCN1	Print stardate message
	LDAB # \$02	Set row number two
	JSR ROWSET	Set up row for printout
	LDAA CQC	Fetch current quadrant contents
	LDX # \$01C3	Set pointer to condition message
	ANDA # \$30	Alien ship in quadrant?
	BNE RED	Yes, condition red
	LDAA # \$C7	No, condition green
	STAA X	Fill in 'GREEN' in
	LDAA # \$D2	Condition message
	STAA \$01,X	
	LDAA # \$C5	
	STAA \$02,X	
	LDAA # \$C5	
	STAA \$03,X	
	LDAA # \$CE	
	STAA \$04,X	
	BRA CND	
SRSCN1	JMP MSG	Output condition message
RED	LDAA # \$D2	Condition red
	STAA X	Fill in 'RED' in
	LDAA # \$C5	Condition message
	STAA \$01,X	
	LDAA # \$C4	
	STAA \$02,X	
	CLR \$03,X	
CND	LDX # \$01B8	Set pointer to condition message
	BSR SRSCN1	Print condition message
	LDAB # \$03	Set row number three
	JSR ROWSET	Set up for printout
	JSR QUAD	Print current quadrant
	LDAB # \$04	Set row number four
	JSR ROWSET	Set up for printout
	LDX # \$01E3	Set up sector message
	STX PNTR1	Pointer in storage
	LDX PSLOSS	Pointer to current sector
	JSR TWO	Put two digits in message
	LDX # \$01D8	Set pointer to sector message
	BSR SRSCN1	Print sector message
	LDAB # \$05	Set row number five
	JSR ROWSET	Set up row for printout
	LDX PDVME	Set pointer to main energy



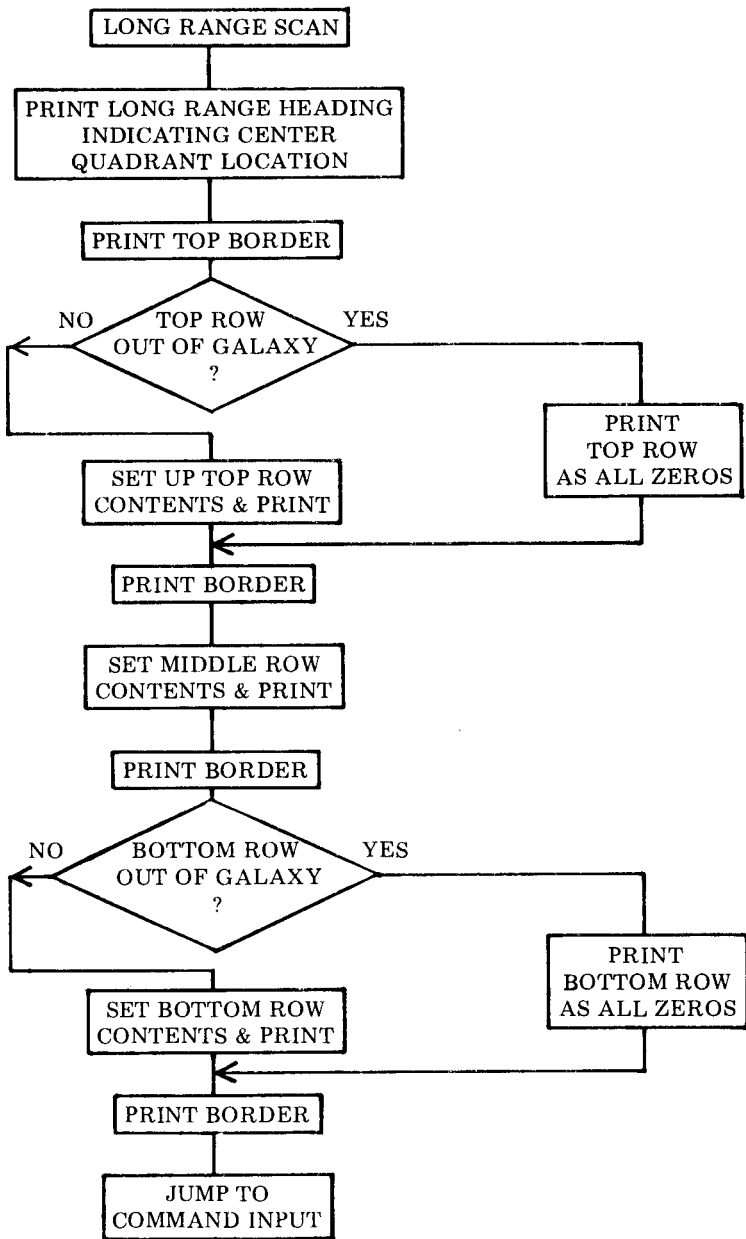
LDAB # \$02	Set precision counter
JSR BINDEC	Convert to decimal
LDX # \$01F5	Message pointer
LDAB # \$04	Counter for four digits
JSR DIGPRT	Put digits in message
LDX # \$01E7	Set pointer to energy message
BSR SRSCN1	Print energy message
LDAB # \$06	Set row number six
JSR ROWSET	Set up for printout
LDX # NTR	Pointer to torpedo count
LDAB # \$01	Precision = 1
JSR BINDEC	Convert to decimal
LDX # \$0203	Set pointer to torpedo message
LDAB # \$02	Counter to number of digits
JSR DIGPRT	Put number of torpedoes in message
LDX # \$01F7	Print torpedo message
JSR SRSCN1	
LDAB # \$07	Set row number seven
JSR ROWSET	Set up row for printout
LDX PDVSE	Set pointer to shield energy
LDAB # \$02	And set precision for
JSR BINDEC	Binary to decimal conversion
LDX # \$0213	Set pointer to shield energy message
LDAB # \$04	Set digit count
JSR DIGPRT	Put digits in memory
LDX # \$0205	Set pointer to shield message
JSR MSG	Print shield message
LDAB # \$08	Set row number eight
JSR ROWSET	Set up row for printout
LDX # \$0170	Set pointer to final row
JSR MSG	Print final row

The commands, input by the operator to direct the operation of the space ship, are controlled by the COMMAND INPUT routine, labeled CMND. This routine (which immediately follows the short range scan) begins by deleting ten units of energy from the main storage bank to simulate the loss of energy resulting from the operation of the ship's control panel. The second byte of the random number storage is then decremented to increase the random number generator's overall randomness. The command request message is then output to the display device, followed by a call to the input routine to receive the command from the input device. If the character input matches one of the ASCII codes (indicating a valid command), the proper routine is entered to perform the command. If the character is not a valid command entry, the program simply requests the command input again. The flow chart and listing for the command input routine are presented next.



CMND	LDX #\$0A00 STX STORE1 JSR ELOM DEC RNM+\$1	Delete ten units of energy For each command Randomize random number
CMD	LDX #\$0215 JSR MSG JSR INPUT	Set pointer to command message Request command input Input command
	CMPA #\$B0 BNE NCRSE JMP CRSE	Ship movement? No, try next Yes, input course
NCRSE	CMPA #\$B1 BNE NSRSCN JMP SRSCN	Short range scan? No, try next Yes, display quadrant
NSRSCN	CMPA #\$B2 BNE NLRSCN JMP LRSCN	Long range scan? No, try next Yes, print long range scan
NLRSCN	CMPA #\$B3 BNE NGXPRT JMP GXPRT	Galaxy printout? No, try next Yes, print galaxy
NGXPRT	CMPA #\$B4 BNE NSHEN JMP SHEN	Shield energy? No, try next Yes, adjust shields
NSHEN	CMPA #\$B5 BNE NPHSR JMP PHSR	Phasor control? No, try next Yes, fire phasors
NPHSR	CMPA #\$B6 BNE CMD JMP TRPD	Torpedo shot? No, illegal command, try again Yes, shoot torpedo

The long range scan routine outputs the contents of the current quadrant and the eight quadrants which immediately surround it. The number of alien ships, space stations, and stars in each of these quadrants is displayed as described in the first chapter. A message is output first indicating the current quadrant location of the space ship. The contents of the three quadrants in the row above the current quadrant are then output by calling the LRR subroutine. If this top row is outside the galaxy, the contents will be output as all zeros by use of the RWC routine. The row containing the current quadrant is then output, followed by the row below the current quadrant. If this bottom row is outside the galaxy, its contents will be displayed as all zeros. A dividing line of dashes is output between each row. At the completion, the routine returns to input a new command. The long range scan routine begins at the label LRSCN. The flow chart and listing for this routine are presented next.



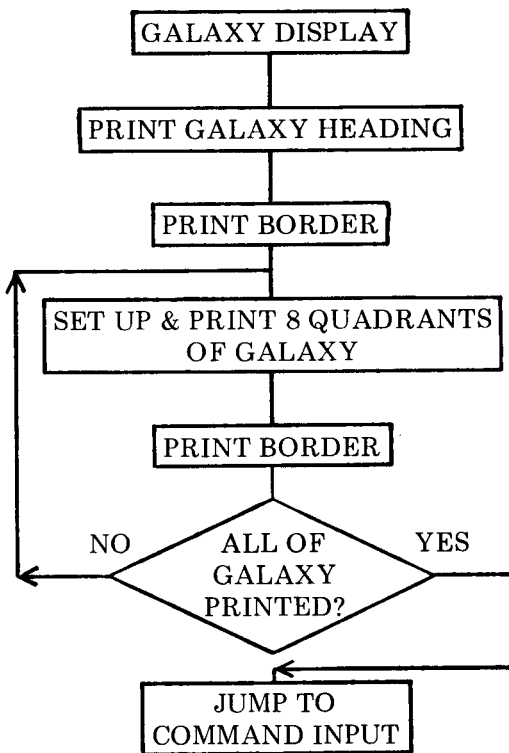
LRSCN	LDX #\$024D JSR MSG JSR QUAD BSR LRSCN1 LDAA CQLSS TAB ANDB #\$38 BEQ RWC1 SUBA #\$08 JSR LRR	Set pointer to long range message Print long range scan Print quadrant location Print row of dashes Fetch current quadrant Save temporarily Current quadrant in row no. 1? Yes, top row clear Indicate row -1 Set up and print top row
LR1	BSR LRSCN1 LDAA CQLSS JSR LRR	Print separating row Fetch current quadrant Set up and print middle row
	BSR LRSCN1 LDAA CQLSS CMPA #\$38 BCC RWC2 ADDA #\$08 JSR LRR	Print separating row Fetch current quadrant Current quadrant in row no. 8? Yes, bottom row clear No, set quadrant row +1 Set and print bottom row
LR2	BSR LRSCN1 JMP CMND	Print bottom border
LRSCN1	JMP NTN	Input next command
RWC1	BSR RWC JMP LR1	Print clear row Continue long range scan
RWC2	BSR RWC JMP LR2	Print clear row Finish long range scan
RWC	LDX #\$04C9 CLRA JSR QDSET LDX #\$04CF CLRA JSR QDSET LDX #\$04D5 CLRA JSR QDSET JMP LRP	Set pointer to left quadrant Set zero entry Set quadrant contents Set pointer to middle quadrant Set zero entry Set quadrant contents Set pointer to right quadrant Set zero entry Set quadrant contents Print long range row

The galaxy display routine produces an output of the entire galaxy contents to the display device in a format similar to that of the long range scan. The display is used to provide the operator with a map from which a course may be charted for the mission. The contents of a complete row are set up in the galaxy printout message on page 00 by calling the QDSET subroutine, and then the row is output to the

display device. A dividing line of dashes is output between each row. When the output is finished, the routine returns to the command input routine. The galaxy display routine flow chart and listing are presented next.

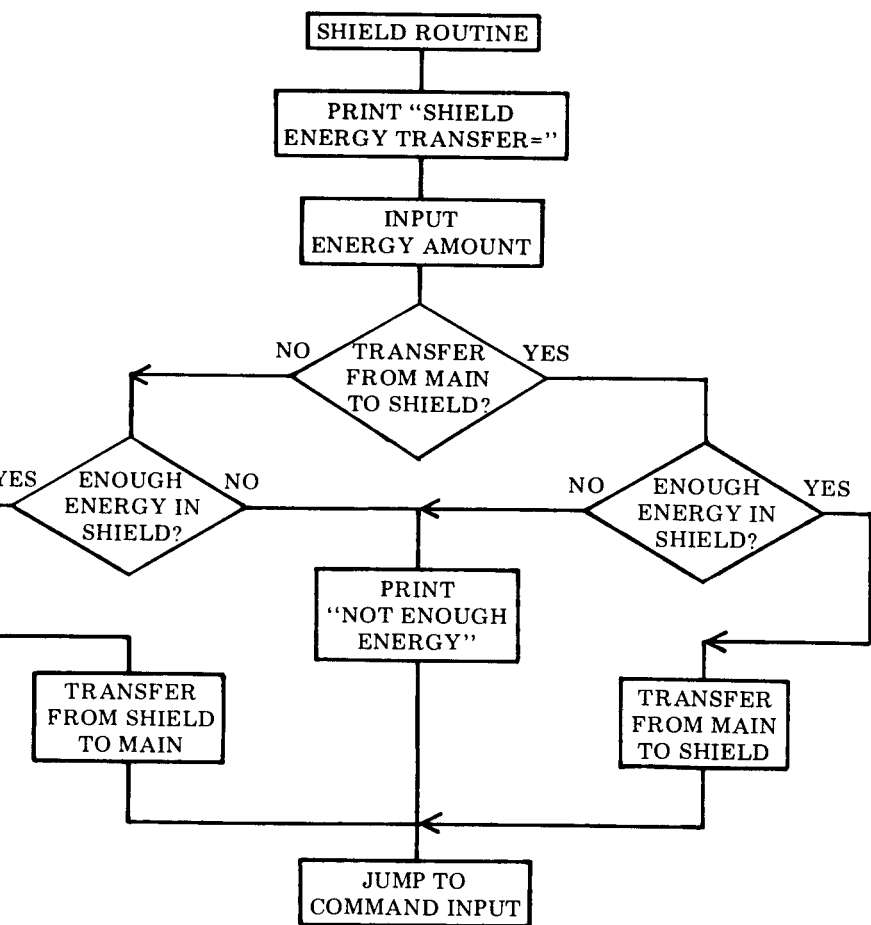
GXPRT	LDX #\$0422	Print GALAXY DISPLAY
	JSR MSG	
	LDAB \$31	
	JSR NT1	Print border
	LDX #\$00C0	Set pointer to galaxy
	STX PNTR1	Store temporarily
GL1	LDX #\$0084	Set up message pointer
	STX PNTR2	Store temporarily
GL2	LDX PNTR1	Fetch galaxy pointer
	CPX #\$0100	End of printout?
	BEQ GL3	Yes, input next command
	LDAA X	Get quadrant contents
	INX	Advance pointer
	STX PNTR1	Restore to memory
	LDX PNTR2	Set up message pointer
	JSR QDSET	Set quadrant contents in MSG
	LDAA #\$06	
	ADDA PNTR2+\$1	Advance message pointer
	STAA PNTR2+\$1	Restore to memory
	CMPA #\$B4	This end of line?
	BNE GL2	No, set next quadrant
	LDX #\$0080	Print current line of galaxy
	JSR MSG	
	LDAB #\$31	
	JSR NT1	Print border
	BRA GL1	Set up next line
GL3	JMP CMND	End, return to command input

The shield routine transfers energy between the main energy supply and the protective shields as designated by the operator. The routine begins by requesting the operator to enter the amount of energy to be transferred. The EIN routine is called to input the energy from the input device. The input is then converted to its binary value and the sign of the input is checked. If a minus sign was



entered preceeding the energy input, the energy is transferred from the shield energy to the main energy storage. If only the four digits are entered, the transfer of energy goes from the main supply to the shields by jumping to the routine labeled POS. In either case, the supply from which the energy is to be taken is checked to determine whether there is enough energy for the transfer. If there is not enough, a message is output to inform the operator, and the routine returns to the command input routine. If there is sufficient energy, the transfer will be completed and the program will return to the command input routine. This routine begins at the label SHEN. The flow chart and listing are presented next.

SHEN	LDX #\$0330	Print SHIELD ENERGY
	JSR MSG	TRANSFER =
	JSR EIN	Input energy amount
	BMI SHEN	Invalid input, try again



JSR DCBN
LDX STORE2
STX STORE1
LDAA DGT5TH
BEQ POS
JSR CKSD
BCS NE
JSR FMDS
JSR TOMN
BRA SHEN1

POS JSR CKMN
BCS NE

Convert to binary
Transfer binary amount for
Routines to follow
Test if have '.' sign
No, transfer main to shields
Check shield energy
Not enough, print message
Subtract from shields
Add to main
Input new command

Check main energy
Not enough, display message

	JSR FMMN	Subtract from main
	JSR TOSD	Add to shields
	BRA SHEN1	Input new command
NE	LDX #\$034C	Print NOT ENOUGH
	JSR MSG	ENERGY
SHEN1	JMP CMND	Input new command

The movement routine is called when it is desired to move the space ship within the galaxy. The course direction is input by calling the DRCT subroutine, which returns with the pointer to the course table stored in the temporary pointer storage labeled PNTR1. The distance, or warp factor, is then entered, and the binary count of the number of sectors to be traversed is stored in the counter storage labeled CNTR. The ACTV subroutine is called to set up the adjusted row and column values used by the TRK subroutine in advancing the space ship. The crossing indicator is cleared before the routine begins the actual movement of the space ship. The crossing indicator is used at the end of the move to indicate whether one or more quadrant borders have been crossed.

Movement of the space ship begins at the label MOV which first calls TRK to move the space ship one sector. If the return from TRK indicates the space ship is outside the known galaxy, the LOST subroutine is called, which ends the current game. Otherwise, a quadrant crossing is checked by reading the crossing flag. If a crossing did not occur, the program checks for a possible collision. However, if the space ship did cross a quadrant border, the crossing indicator is set, 25 units of energy are deleted from the main supply, and the new quadrant is set up.

The routine then checks for a collision between the space ship and the other objects in the quadrant. If a collision occurs within the initial quadrant, the result will be one of the following. For a collision with a star, the game will end by jumping to the WPOUT subroutine. A collision with a space station results in the elimination of the space station and the loss of 600 units of energy from the ship's shields. Finally, a collision with an alien ship results in its elimination, and a loss of 1500 units of energy from the space ship's shields.

After a collision with a space station or alien ship, or if there was

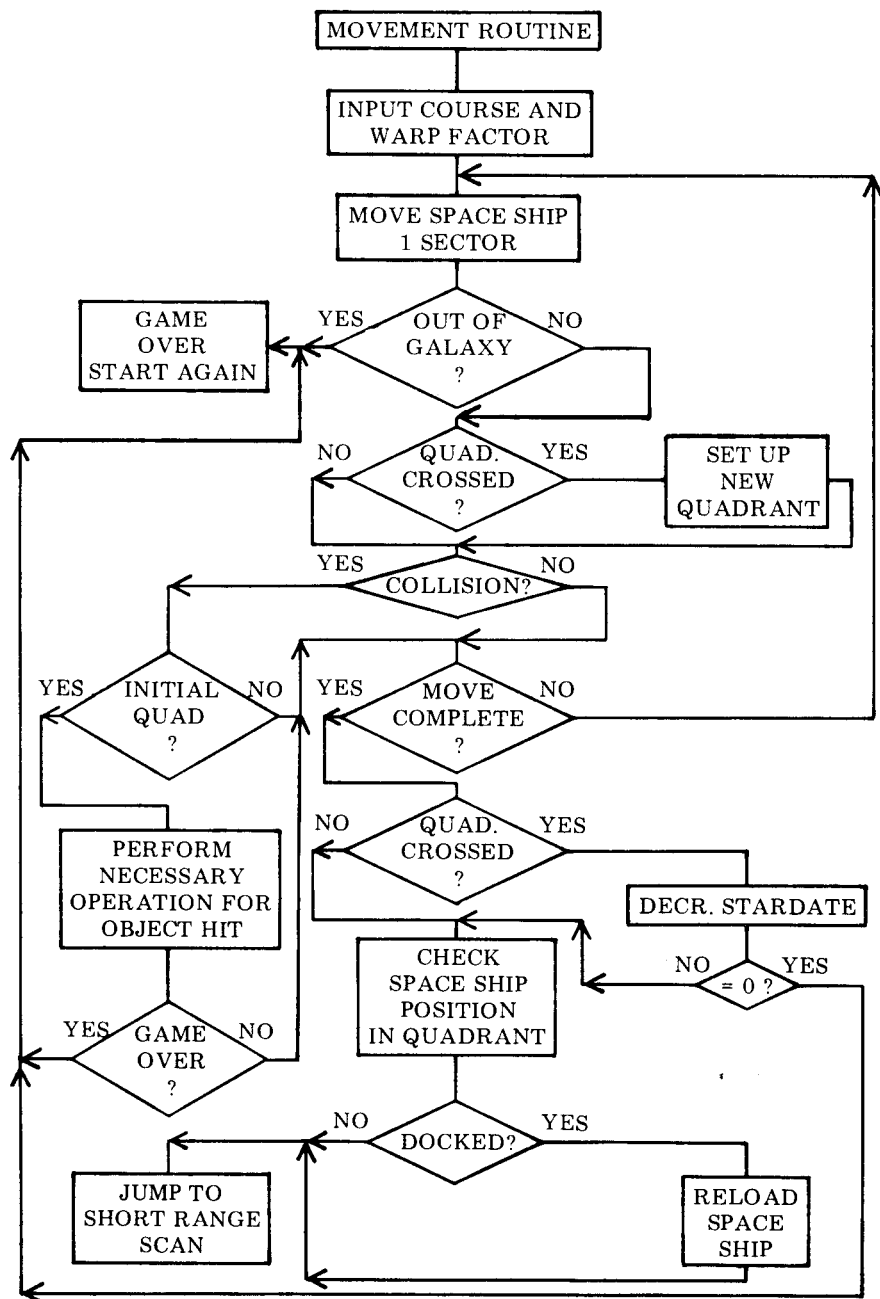
no collision, the move is continued by decrementing the warp factor and, if not zero, returning to MOV to move the space ship one more sector. When the warp factor reaches zero, the crossing indicator is checked, and if set, the stardate counter is decremented. When the stardate counter goes to zero, the operator has run out of time and the game ends by jumping to the TIME subroutine.

The location of the space ship is then checked against the location of the other objects in the quadrant. If the space ship is in the same sector as another object in the quadrant, the other object is moved. This coincidence may occur when the space ship moves into a new quadrant, since a collision outside the original quadrant is ignored in the collision routine.

The final operation of this routine is to check for a docking with a space station. This can only occur when the space ship completes its move by residing in a sector on either side of the space station. The space ship is not docked when it is in the sector above or below the space station. If the space ship is docked, its energy banks and torpedo tubes are refilled. The flow chart and listing for the movement routine are now presented.

CRSE	LDX #\$0220	Set pointer to course message
	JSR MSG	Request course input
	JSR DRCT	Input course direction
	BEQ CRSE	Input error, try again
WRP	LDX #\$0233	Index to WARP message
	JSR MSG	Request warp input
	JSR INPUT	Input warp factor digit 1
	CMPA #\$B0	Is digit less than 0?
	BCS WRP	Yes, request input again
	CMPA #\$B8	Is digit greater than 7?
	BCC WRP	Yes, try again
	ANDA #\$07	Mask off ASCII code
	ASLA	Position to 3rd bit
	ASLA	
	ASLA	
	TAB	Store temporarily in B
	LDAA #\$AE	Print decimal point
	JSR PRINT	
	JSR INPUT	Input 2nd warp factor digit
	CMPA #\$B0	Is digit less than 0?
	BCS WRP	Yes, request input again
	CMPA #\$B8	Is input greater than 7?

	BCC WRP ANDA #\$07 ABA BEQ WRP STAA CNTR JSR ACTV CLR CI	Yes, no good, try again Mask off ASCII code Add warp digit 1 If 0, no good, try again Store warp factor as counter Fetch adjusted row and column Clear crossing indicator
MOV	JSR TRK BNE MOV1 JMP LOST	Track one sector Out of galaxy? No Yes, lost in space
MOV1	LDAA CF BEQ CLSN STAA CI LDX #\$1900 STX STORE1 JSR ELOM JSR QCNT JSR NWQD	No, quadrant crossed? No, check for collision Make crossing indicator non-zero Delete 25 units of energy From main supply Fetch new quadrant contents Set up new quadrant
CLSN	JSR RWCM JSR MATCH BNE MVDN JSR COMPAR BEQ SSOUT BCC ASOUT LDAA CI BNE MVDN JMP WPOUT	Form row and column byte Collision? No, complete move What was hit? Space ship collision! Alien ship collision! Star, initial quadrant? No, ignore collision Yes, ship wiped out!
MVDN	DEC CNTR BNE MOV LDAA CI BEQ NOX DEC NSR BNE NOX JMP TIME	Decrement warp factor Not zero, continue move Fetch crossing indicator Quadrant not crossed, continue move Decrement stardate counter Not zero, continue Ran out of time, start new game
NOX	JSR RWCM STAA SLOSS JSR MATCH BNE NOX1 JSR CHNG	Form row and column byte Save new sector Was last move a collision? No, check for docking Yes, change object location
NOX1	JSR DKED JMP SRSCN	Check for docking Do short range scan
SSOUT	LDAA CI BNE MVDN JSR DLET	Test if initial quadrant No, no loss Remove space station from galaxy

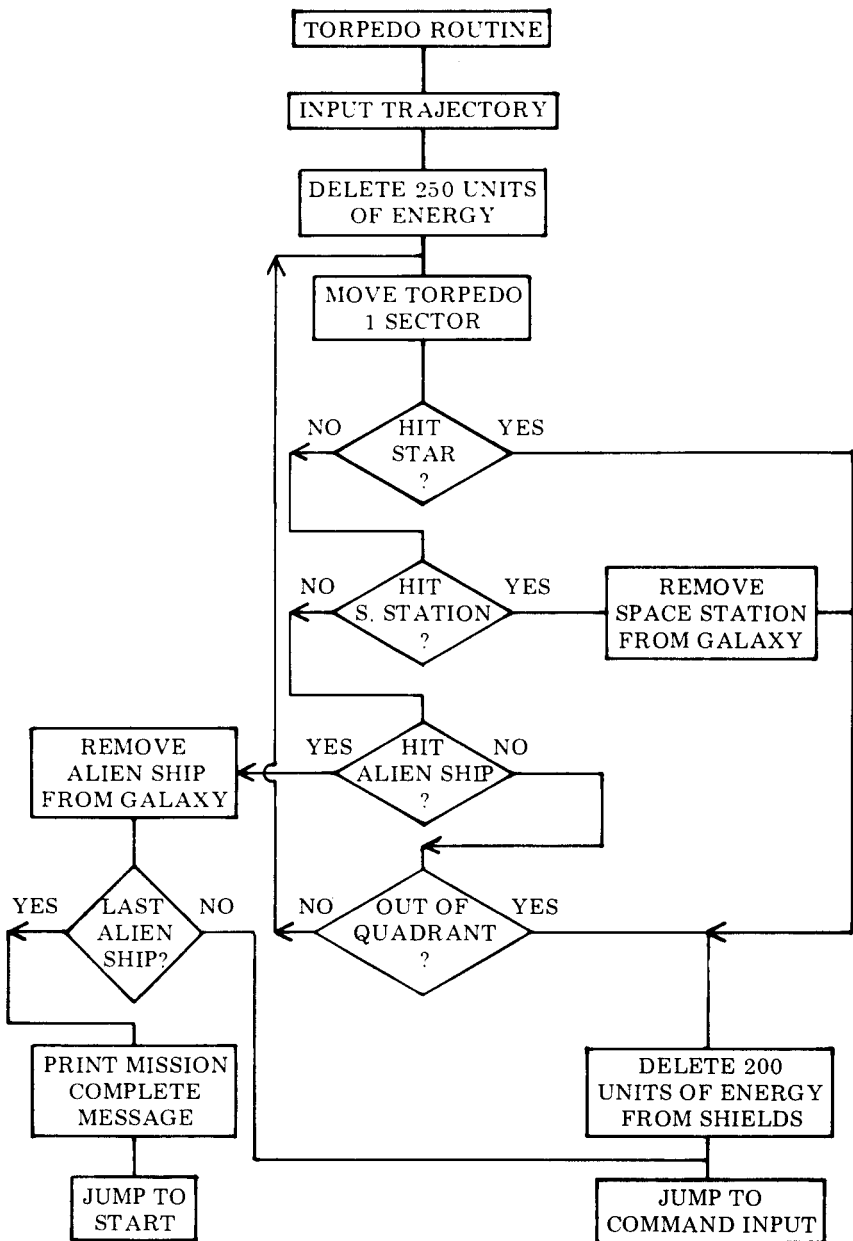


	LDX #\$03BA	Indicate loss of space station
	JSR MSG	
	LDX #\$5802	Then delete 600 units
	STX STORE1	Of energy from sheilds
SSO1	JSR ELOS	Delete energy
	JMP MVDN	Finish move
ASOUT	LDAA CI	Test if initial quadrant
	BNE MVDN	No, no loss
	JSR DLET	Yes, delete alien ship
	LDX #\$037F	Print alien ship destroyed message
	JSR MSG	
	LDX #\$DC05	Delete 1500 units of
	STX STORE1	Energy from space ship
	BRA SSO1	
CHNG	LDAB #\$01	Set number of objects counter
	JMP LOCSET	Move object and return
DKED	LDAA SLSS	Is space station in quadrant?
	BPL DKED1	Yes, continue
	RTS	No, complete move
DKED1	ANDA #\$38	Mask out row
	LDAB SLOSS	Fetch space ship location
	ANDB #\$38	Mask out row
	CBA	Same row?
	BNE DKED2	No, return
	LDAA SLSS	Fetch space station location
	LDAB SLOSS	Fetch space ship location
	ADDB #\$01	Docked on right?
	CBA	
	BEQ DKED3	Yes, reload
	SUBB #\$02	No, check left docking
	CBA	Docked on left?
	BEQ DKED3	Yes, reload
DKED2	RTS	No, return
DKED3	JMP LOAD	Reload space ship and return

The torpedo routine fires a torpedo in the direction specified by the operator in an attempt to destroy an alien ship. This routine first checks the number of torpedoes available. If there are no torpedoes remaining, a message in output to inform the operator, and the routine returns to the command routine. If there is a torpedo available, the torpedo count is decremented, and 250 units of energy are depleted from the main storage bank. The DRCT subroutine is then called to input the direction of fire for the torpedo. The ACTV subroutine then sets the adjusted row and column values for tracking the torpedo.

Once the trajectory is set up, the torpedo is moved one sector at a time, using the TRK subroutine. If the torpedo moves out of the quadrant, it has missed its intended target and the alien ship retaliates by firing 200 units of phasor energy back at the space ship. Otherwise, the sector location of the torpedo is output in the tracking message so that the operator can follow the torpedo's path. The MATCH subroutine checks for a collision after each sector moved. If there is no collision at this sector, the torpedo will be tracked another sector by returning to the TR2 label in this routine. If an alien ship has been hit, it is removed from the galaxy. If it is the last alien ship, the mission is complete, and the program begins a new game. If a space station is hit, it is eliminated and the alien ship will retaliate as mentioned above. If a star is hit, the torpedo has missed its mark and the alien ship will again retaliate for the attempted attack. The program then returns to the command input routine. The torpedo flow chart and listing are presented next.

TRPD	LDAA NTR	Any torpedoes left?
	BEQ NTPD	No, print no torpedo message
	DEC NTR	Yes, delete one
	LDX #\$FA00	Setup 250 units
	STX STORE1	Of energy to delete
	JSR CKMN	Enough in main supply?
	BCC TRPD1	Yes, continue
	JMP NE	No, report not enough energy
TRPD1	JSR FMMN	Delete from main
TR1	LDX #\$0360	Print 'TORPEDO TRAJECTORY ='
	BSR TR3	
	JSR DRCT	Input direction
	BEQ TR1	Input invalid, try again
	JSR ACTV	Form adjusted row and column
	LDAA CQLSS	Save current quadrant
	STAA CNTR	Location in temporary storage
TR2	JSR TRK	Move torpedo one sector
	BEQ QOUT	Out of galaxy? Yes, missed
	LDAA CF	Quadrant crossed?
	BNE QOUT	Yes, missed
	JSR RWCM	No, form row and column byte
	TAB	
	STAA STORE1	Move to temporary storage
	LDX #\$041E	Set up tracking message
	JSR T1	Print TRACKING: R,C
	LDX #\$0412	Form message pointer
	BSR TR3	Print message



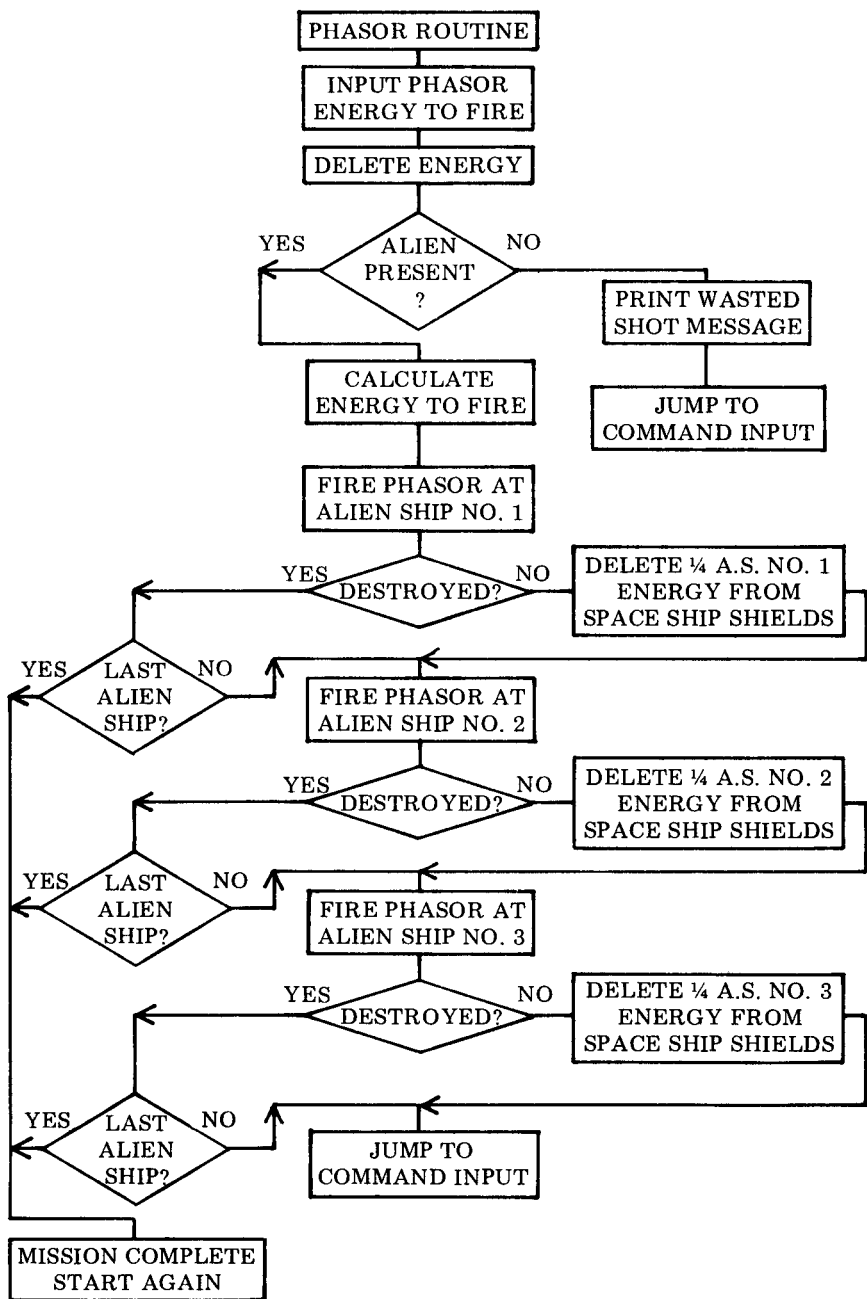
	LDAA STORE1	Fetch row and column byte
	JSR MATCH	Torpedo hit anything?
	BEQ HIT	Yes, analyze
	JMP TR2	No, continue tracking
HIT	JSR COMPAR	What was hit? A star?
	BCS QOUT	Yes, missed alien ship
	BEQ SSTA	Space station? Yes, delete space station
	JSR DLET	No, delete alien ship
	LDX #\$037F	Print alien ship hit message
	BSR TR3	
	BRA CMND1	Input new command
TR3	JMP MSG	Print message and return
SSTA	JSR DLET	Delete space station from galaxy
	LDX #\$03BA	Print message of loss of
	BSR TR3	Space station
QOUT	LDAA CNTR	Restore current quadrant location
	STAA CQLSS	of the space ship
	JSR WASTE	See if any alien ships in quadrant
	LDX #\$0396	
	JSR MSG	No, print missed message
	LDX #\$C800	Set up loss of 200 units of energy
	JSR ELOS	Due to alien ship retaliating
	BRA CMND1	Input new command
NTPD	LDX #\$04B6	Print no torpedo message
	JSR MSG	
CMND1	JMP CMND	Input new command
WASTE	LDAA CQC	Fetch quadrant contents
	ANDA #\$30	Mask out alien ship count
	BEQ WASTE1	If none, wasted shot
	RTS	Otherwise, return
WASTE1	PULB	Remove unwanted address
	PULB	From stack
	LDX #\$0479	Set pointer to wasted shot message
	JSR MSG	Print message
	JMP CMND	Input new command

The phasor routine fires a designated amount of phasor energy at the alien ships in the quadrant. The EIN subroutine is called to input the energy to be fired. The amount of energy entered is then deleted from the main storage bank. The number of alien ships in the immediate quadrant is then determined to calculate the amount of energy to be fired at each. If there are no alien ships, a message is output indicating the energy fired was wasted. The amount of phasor

energy to be fired at the alien ships is calculated and saved for use by the ASPH subroutine.

The ASPH subroutine is called to fire the phasor at each of the three possible alien ships in the quadrant. It first ascertains the presence of the particular alien ship by looking for its row and column location in the data table. If this location contains a C0, no alien ship is located here and the routine simply returns. Otherwise, this row and column location is output to inform the operator which alien ship is about to be attacked. The distance between the space ship and the alien ship, as defined in Chapter One, is then calculated and the distance factor is used to determine how much of the phasor energy actually reaches the alien ship. This energy is subtracted from the alien ship's shield energy, and if the result is zero or less, the alien ship is destroyed. A message is output to inform the operator of its destruction. If the alien ship is not destroyed, the new energy level of the alien ship's shields is output and, in retaliation, the alien ship fires a phasor equal to one quarter of its shield energy at the space ship. When the ASPH subroutine has completed its operation, it returns to the phasor routine. After all alien ships in the quadrant have been fired upon, the phasor routine returns to the command input routine. The phasor routine flow chart and listing is now presented.

PHSR	LDX #\$0433	Print 'PHASOR ENERGY TO FIRE='
	BSR TR3	
	JSR EIN	Input energy amount
	BMI PHSR	Input error? Try again
	JSR DCBN	Convert decimal to binary
	LDX STORE2	Move binary energy value to proper
	STX STORE1	Storage for ELOM routine
	JSR ELOM	Delete energy from main supply
	JSR WASTE	Check for presence of alien ships
PHS1	JSR ROTR4	Position alien ship number
	SUBA #\$01	1 alien ship, full energy
	BEQ PH1	2 alien ships, half energy
	TAB	3 alien ships, 1/4 energy
	JSR DVD	Divide energy accordingly
PH1	LDX STORE1	Fetch energy amount
	STX STORE4	Save energy amount
	LDX PVASE1	Fetch pointer to alien ship no. 1 energy
	STX PNTR3	Save pointer for ASPH routine
	LDX PSLAS1	Pointer to alien ship no. 1 position
	JSR ASPH	Fire phasor at alien ship no. 1
	LDX PVASE2	Fetch pointer to alien ship no. 2 energy



	STX PNTR3	Save pointer for ASPH routine
	LDX PSLAS2	Pointer to alien ship no. 2 position
	JSR ASPH	Fire phasor at alien ship no. 2
	LDX PVASE3	Fetch pointer to alien ship no. 3 energy
	STX PNTR3	Save pointer for ASPH routine
	LDX PSLAS3	Pointer to alien ship no. 3 position
	JSR ASPH	Fire phasor at alien ship no. 3
	BRA CMND1	Input new command
ASPH	STX PNTR2	Save position pointer
	LDAA X	Fetch alien ship location
	BPL ASPH1	Any alien ship in location?
	RTS	No, return
ASPH1	LDX STORE4	Restore energy value
	STX STORE1	Move to temporary storage
	LDX #\$0465	Set up pointers
	STX PNTR1	To fill in alien ship location
	LDX PNTR2	In message
	JSR TWO	Set sector coordinates
	LDX #\$044E	Print 'ALIEN SHIP AT SECTOR X,Y:'
	JSR MSG	
	LDX #SLOSS	Fetch sector location of the space ship
	BSR SPRC	Separate row and column values
	STAA STORE2	Save row of space ship
	STAA STORE2+\$1	Save column of space ship
	LDX PNTR2	Fetch pointer to alien ship location
	BSR SPRC	Separate row and column values
	SUBA STORE2	Create row difference
	BPL PH2	Make absolute difference
	NEGA	By negating a negative value
PH2	SUBB STORE2+\$1	Create column difference
	BPL PH3	Make absolute difference
	NEGB	By negating a negative value
PH3	ABA	Add absolute differences
	LSRA	Divide by 4 to
	LSRA	Form the distance factor
	ANDA #\$03	Of energy to reach alien ship
	TAB	Store in B
	BEQ PH4	Make sure not zero
	JSR DVD	Calculate energy that reached alien ship
PH4	LDX PNTR3	Subtract from shield energy
	JSR FM1	Of alien ship
	BMI DSTR	If negative, alien ship is destroyed
	BNE ALOS	If non-zero, print alien ship energy
	TST X	Alien ship energy = 0?
	BEQ DSTR	Yes, remove from galaxy
ALOS	LDAB #\$02	Set precision counter
	JSR BINDEC	Convert alien ship energy to decimal
	LDX #\$0477	Set digits in message

	LDAB #\$04	Set number of digits counter
	JSR DIGPRT	Put digits in message
	LDX #\$046B	Print energy of alien ship
	JSR MSG	
	LDX PNTR3	Set pointer to alien ship energy
	LDAA X	Transfer alien ship energy
	STAA STORE1	To STORE1 for calculating
	LDAA \$01,X	Retaliation amount
	STAA STORE1+\$01	
	LDAB #\$02	Divide energy by 4 as
	JSR DVD	Retaliation by alien ship
	LDX STORE1	Place energy into index register
	JMP ELOS	Remove from shield energy, return
DSTR	LDX #\$03CA	Print 'DESTROYED'
	JSR MSG	
	LDX PNTR2	Fetch alien ship location
	JMP DLET	Remove alien ship from galaxy, return
SPRC	LDAA X	Fetch row and column byte
	TAB	Save for column value
	JSR ROTR3	Position row to right
	ANDA #\$07	Mask out row value
	ANDB #\$07	Mask out column value
	RTS	Return

6800 ASSEMBLED LISTING

This chapter contains the assembled listing for the 6800 Galaxy program. The assembled listing provides the memory addresses and machine code for the mnemonics which make up the Galaxy program. All that is required is to add the reader provided I/O driver routines for the specific devices available on one's system. These routines must follow the guidelines described in Chapter Two. For systems that use the MIKBUG** program for I/O, sample routines for input and output via MIKBUG** are presented at the end of this listing.

The first portion of the listing indicates the usage of page 00 for the course table, temporary data storage, the galaxy display message, and the galaxy content table. The galaxy display message on page 00, the messages of page 01 through 04, and the galaxy setup table on page 0F are presented as octal dumps.

The start of execution address for the Galaxy program as presented herein is page 05 location 00.

0000	02	\$02	Course 1.0
0001	00	\$00	
0002	02	\$02	Course 1.5
0003	FF	\$FF	
0004	02	\$02	Course 2.0
0005	FE	\$FE	
0006	01	\$01	Course 2.5
0007	FE	\$FE	
0008	00	\$00	Course 3.0
0009	FE	\$FE	
000A	FF	\$FF	Course 3.5
000B	FE	\$FE	
000C	FE	\$FE	Course 4.0
000D	FE	\$FE	
000E	FE	\$FE	Course 4.5
000F	FF	\$FF	
0010	FE	\$FE	Course 5.0
0011	00	\$00	
0012	FE	\$FE	Course 5.5
0013	01	\$01	
0014	FE	\$FE	Course 6.0
0015	02	\$02	
0016	FF	\$FF	Course 6.5
0017	02	\$02	

	0018	00	\$00		Course 7.0
	0019	02	\$02		
	001A	01	\$01		Course 7.5
	001B	02	\$02		
	001C	02	\$02		Course 8.0
	001D	02	\$02		
	001E	02	\$02		Course 8.5
	001F	01	\$01		
0020		PNTR1	RMB \$2		Temp pointer storage area
0022		PNTR2	RMB \$2		
0024		PNTR3	RMB \$2		
0026		STORE1	RMB \$2		Temp data storage area
0028		STORE2	RMB \$2		
002A		STORE3	RMB \$2		
002C		STORE4	RMB \$2		
002E		STORE5	RMB \$2		
0030		CNTR	RMB \$1		Temporary counter storage
0031		CI	RMB \$1		Crossing indicator
0032		CF	RMB \$1		Crossing flag
0033		RNM	RMB \$2		Random number storage
0035		CQC	RMB \$1		Current quadrant's contents
0036		SLOSS	RMB \$1		Sector location of space ship
0037		SOLSS	RMB \$7		Sector location of stars
003E		SLSS	RMB \$1		Sector location of space station
003F		SLAS1	RMB \$1		Sect. loc. of alien ship no. 1
0040		SLAS2	RMB \$1		Sect. loc. of alien ship no. 2
0041		SLAS3	RMB \$1		Sect. loc. of alien ship no. 3
0042		DVME	RMB \$2		Energy in main supply
0044		DVSE	RMB \$2		Energy in shields
0046		VASE1	RMB \$2		Alien ship no. 1 energy
0048		VASE2	RMB \$2		Alien ship no. 2 energy
004A		VASE3	RMB \$2		Alien ship no. 3 energy
004C		CQLSS	RMB \$1		Quadrant loc. of space ship
004D		NTR	RMB \$1		Number of torpedoes
004E		NSS	RMB \$1		Number of space stations
004F		NAS	RMB \$1		Number of alien ships
0050		NSR	RMB \$1		Number of star dates left
0051		DGT1ST	RMB \$1		Digit storage for
0052		DGT2ND	RMB \$1		Binary to decimal and
0053		DGT3RD	RMB \$1		Decimal to binary
0054		DGT4TH	RMB \$1		Conversion
0055		DGT5TH	RMB \$1		
0056	00	PSTR1	\$00		Table of pointers
0057	26		\$26		Used for loading
0058	00	PSTR51	\$00		The index register
0059	2F		\$2F		

005A	00	PSLOSS	\$00
005B	36		\$36
005C	00	PSOLSS	\$00
005D	37		\$37
005E	00	PSLSS	\$00
005F	3E		\$3E
0060	00	PSLAS1	\$00
0061	3F		\$3F
0062	00	PSLAS2	\$00
0063	40		\$40
0064	00	PSLAS3	\$00
0065	41		\$41
0066	00	PDVME	\$00
0067	42		\$42
0068	00	PDVSE	\$00
0069	44		\$44
006A	00	PVASE1	\$00
006B	46		\$46
006C	00	PVASE2	\$00
006D	48		\$48
006E	00	PVASE3	\$00
006F	4A		\$4A
0070	00	PCQLSS	\$00
0071	4C		\$4C
0072	00	PDG1ST	\$00
0073	51		\$51
0074	00	PDG5TH	\$00
0075	55		\$55

0080	8D	8A	B1	A0	B0	B0	B0	A0
0088	B1	A0	B0	B0	B0	A0	B1	A0
0090	B0	B0	B0	A0	B1	A0	B0	B0
0098	B0	A0	B1	A0	B0	B0	B0	A0
00A0	B1	A0	B0	B0	B0	A0	B1	A0
00A8	B0	B0	B0	A0	B1	A0	B0	B0
00B0	B0	A0	B1	00				

00C0 through 00FF reserved for Galaxy Content Table

0100	8D	8A	C4	CF	A0	D9	CF	D5
0108	A0	D7	C1	CE	D4	A0	D4	CF
0110	A0	C7	CF	A0	CF	CE	A0	C1
0118	A0	D3	D0	C1	C3	C5	A0	D6
0120	CF	D9	C1	C7	C5	BF	A0	00
0128	8D	8A	D9	CF	D5	A0	CD	D5
0130	D3	D4	A0	C4	C5	D3	D4	D2
0138	CF	D9	A0	B2	B4	A0	C1	CC

0140	C9	C5	CE	A0	D3	C8	C9	D0
0148	D3	A0	C9	CE	A0	B2	B9	A0
0150	D3	D4	C1	D2	C4	C1	D4	C5
0158	D3	A0	D7	C9	D4	C8	A0	B5
0160	A0	D3	D0	C1	C3	C5	A0	D3
0168	D4	C1	D4	C9	CF	CE	D3	00
0170	8D	8A	A0	AD	B1	AD	AD	B2
0178	AD	AD	B3	AD	AD	B4	AD	AD
0180	B5	AD	AD	B6	AD	AD	B7	AD
0188	AD	B8	AD	00	8D	8A	B8	A0
0190	A0	A0	A0	A0	A0	A0	A0	A0
0198	A0	A0	A0	A0	A0	A0	A0	A0
01A0	A0	A0	A0	A0	A0	A0	A0	00
01A8	A0	D3	D4	C1	D2	C4	C1	D4
01B0	C5	A0	A0	B3	B0	B2	B1	00
01B8	A0	C3	CF	CE	C4	C9	D4	C9
01C0	CF	CE	A0	C7	D2	C5	C5	CE
01C8	00	A0	D1	D5	C1	C4	D2	C1
01D0	CE	D4	A0	A0	B5	AC	B8	00
01D8	A0	D3	C5	C3	D4	CF	D2	A0
01E0	A0	A0	A0	B6	AC	B1	00	A0
01E8	C5	CE	C5	D2	C7	D9	A0	A0
01F0	A0	A0	B5	B0	B0	B0	00	A0
01F8	D4	CF	D2	D0	C5	C4	CF	C5
0200	D3	A0	B1	B0	00	A0	D3	C8
0208	C9	C5	CC	C4	D3	A0	A0	A0
0210	B0	B0	B0	B0	00	8D	8A	C3
0218	CF	CD	CD	C1	CE	C4	BF	00
0220	8D	8A	C3	CF	D5	D2	D3	C5
0228	A0	A8	B1	AD	B8	AE	B5	A9
0230	BF	A0	00	8D	8A	D7	C1	D2
0238	D0	A0	C6	C1	C3	D4	CF	D2
0240	A0	A8	B0	AE	B1	AD	B7	AE
0248	B7	A9	BF	A0	00	8D	8A	CC
0250	AE	D2	AE	A0	D3	C3	C1	CE
0258	A0	C6	CF	D2	00	8D	8A	CD
0260	C9	D3	D3	C9	CF	CE	A0	C6
0268	C1	C9	CC	C5	C4	AC	A0	D9
0270	CF	D5	A0	C8	C1	D6	C5	A0
0278	D2	D5	CE	A0	CF	D5	D4	A0
0280	CF	C6	A0	D3	D4	C1	D2	C4
0288	C1	D4	C5	D3	00	8D	8A	CB
0290	C1	AD	C2	CF	CF	CD	AC	A0
0298	D9	CF	D5	A0	C3	D2	C1	D3
02A0	C8	C5	C4	A0	C9	CE	D4	CF
02A8	A0	C1	A0	D3	D4	C1	D2	AE
02B0	A0	D9	CF	D5	D2	A0	D3	C8
02B8	C9	D0	A0	C9	D3	A0	C4	C5
02C0	D3	D4	D2	CF	D9	C5	C4	00
02C8	8D	8A	D9	CF	D5	A0	CD	CF
02D0	D6	C5	C4	A0	CF	D5	D4	A0

02D8	CF	C6	A0	D4	C8	C5	A0	C7
02E0	C1	CC	C1	D8	D9	AC	A0	D9
02E8	CF	D5	D2	A0	D3	C8	C9	D0
02F0	A0	C9	D3	A0	CC	CF	D3	D4
02F8	AE	AE	CC	CF	D3	D4	00	8D
0300	8A	CC	CF	D3	D3	A0	CF	C6
0308	A0	C5	CE	C5	D2	C7	D9	A0
0310	B0	B1	B2	B9	00	8D	8A	C4
0318	C1	CE	C7	C5	D2	AD	D3	C8
0320	C9	C5	CC	C4	A0	C5	CE	C5
0328	D2	C7	D9	A0	B0	B0	B0	00
0330	8D	8A	D3	C8	C9	C5	CC	C4
0338	A0	C5	CE	C5	D2	C7	D9	A0
0340	D4	D2	C1	CE	D3	C6	C5	D2
0348	A0	BD	A0	00	8D	8A	CE	CF
0350	D4	A0	C5	CE	CF	D5	C7	C8
0358	A0	C5	CE	C5	D2	C7	D9	00
0360	8D	8A	D4	CF	D2	D0	C5	C4
0368	CF	A0	D4	D2	C1	CA	C5	C3
0370	D4	CF	D2	D9	A8	B1	AD	B8
0378	AE	B5	A9	A0	BA	A0	00	8D
0380	8A	C1	CC	C9	C5	CE	A0	D3
0388	C8	C9	D0	A0	C4	C5	D3	D4
0390	D2	CF	D9	C5	C4	00	8D	8A
0398	D9	CF	D5	A0	CD	C9	D3	D3
03A0	C5	C4	A1	A0	C1	CC	C9	C5
03A8	CE	A0	D3	C8	C9	D0	A0	D2
03B0	C5	D4	C1	CC	C9	C1	D4	C5
03B8	D3	00	8D	8A	D3	D0	C1	C3
03C0	C5	A0	D3	D4	C1	D4	C9	CF
03C8	CE	A0	C4	C5	D3	D4	D2	CF
03D0	D9	C5	C4	00	8D	8A	C3	CF
03D8	CE	C7	D2	C1	D4	D5	CC	C1
03E0	D4	C9	CF	CE	D3	AC	A0	D9
03E8	CF	D5	A0	C8	C1	D6	C5	A0
03F0	C5	CC	C9	CD	C9	CE	C1	D4
03F8	C5	C4	A0	C1	CC	CC	A0	CF
0400	C6	A0	D4	C8	C5	A0	C1	CC
0408	C9	C5	CE	A0	D3	C8	C9	D0
0410	D3	00	8D	8A	D4	D2	C1	C3
0418	CB	C9	CE	C7	BA	A0	B3	AC
0420	B3	00	8D	8A	C7	C1	CC	C1
0428	D8	D9	A0	C4	C9	D3	D0	CC
0430	C1	D9	00	8D	8A	D0	C8	C1
0438	D3	CF	D2	A0	C5	CE	C5	D2
0440	C7	D9	A0	D4	CF	A0	C6	C9
0448	D2	C5	A0	BD	A0	00	8D	8A
0450	C1	CC	C9	C5	CE	A0	D3	C8
0458	C9	D0	A0	C1	D4	A0	D3	C5
0460	C3	D4	CF	D2	A0	B7	AC	B8
0468	BA	A0	00	C5	CE	C5	D2	C7

0470	D9	A0	BD	A0	B0	B5	B1	B8
0478	00	8D	8A	CE	CF	A0	C1	CC
0480	C9	C5	CE	A0	D3	C8	C9	D0
0488	D3	A1	A0	D7	C1	D3	D4	C5
0490	C4	A0	D3	C8	CF	D4	00	8D
0498	8A	C1	C2	C1	CE	C4	CF	CE
04A0	A0	D3	C8	C9	D0	A1	A0	CE
04A8	CF	A0	C5	CE	C5	D2	C7	D9
04B0	A0	CC	C5	C6	D4	00	8D	8A
04B8	CE	CF	A0	D4	CF	D2	D0	C5
04C0	C4	CF	C5	D3	00	8D	8A	B1
04C8	A0	B0	B0	B0	A0	B1	A0	B0
04D0	B0	B4	A0	B1	A0	B0	B0	B0
04D8	A0	B1	00	8D	8A	CC	C1	D3
04E0	D4	00	8D	8A	C3	C8	C9	C3
04E8	CB	C5	CE	A1	00			

0500	8E	0E	FF		GALAXY	LDS #0EFF
0503	7E	09	72			JMP START
0506	A6	00			MSG	LDAA X
0508	27	06				BEQ MSG1
050A	BD	0F	C0			JSR PRINT
050D	08					INX
050E	20	F6				BRA MSG
0510	39				MSG1	RTS
0511	47				ROTR4	ASRA
0512	47				ROTR3	ASRA
0513	47					ASRA
0514	47					ASRA
0515	39					RTS
0516	96	33			RN	LDAA RNM
0518	49					ROLA
0519	98	33				EORA RNM
051B	46					RORA
051C	7C	00	34			INC RNM+\$1
051F	9B	34				ADDA RNM+\$1
0521	28	03				BVC SKIP
0523	7A	00	34			DEC RNM+\$1
0526	97	33			SKIP	STAA RNM
0528	39					RTS
0529	DF	20			BINDEC	STX PNTR1
052B	DE	72				LDX PDG1ST
052D	6F	00				CLR X
052F	6F	01				CLR \$01,X
0531	6F	02				CLR \$02,X

0533	6F	03		CLR \$03,X
0535	6F	04		CLR \$04,X
0537	DE	20		LDX PNTR1
0539	A6	00		LDAA X
053B	5A			DECB
053C	27	02		BEQ BNDC
053E	E6	01		LDAB \$01,X
0540	97	26		BNDC STAA STORE1
0542	D7	27		STAB STORE1+\$1
0544	CE	10	27	LDX #\$1027
0547	DF	28		STX STORE2
0549	8D	21		BSR BD
054B	D7	55		STAB DGT5TH
054D	CE	E8	03	LDX #\$E803
0550	DF	28		STX STORE2
0552	8D	18		BSR BD
0554	D7	54		STAB DGT4TH
0556	CE	64	00	LDX #\$6400
0559	DF	28		STX STORE2
055B	8D	0F		BSR BD
055D	D7	53		STAB DGT3RD
055F	86	0A		LDAA #\$0A
0561	97	28		STAA STORE2
0563	8D	07		BSR BD
0565	D7	52		STAB DGT2ND
0567	96	26		LDAA STORE1
0569	97	51		STAA DGT1ST
056B	39			RTS
056C	5F			BD CLR B
056D	5C			BD1 INCB
056E	96	26		LDAA STORE1
0570	90	28		SUBA STORE2
0572	97	26		STAA STORE1
0574	96	27		LDAA STORE1+\$1
0576	92	29		SBCA STORE2+\$1
0578	97	27		STAA STORE1+\$1
057A	24	F1		BCC BD1
057C	96	26		LDAA STORE1
057E	9B	28		ADDA STORE2
0580	97	26		STAA STORE1
0582	96	27		LDAA STORE1+\$1
0584	99	29		ADCA STORE2+\$1
0586	97	27		STAA STORE1+\$1
0588	5A			DECB
0589	39			RTS
058A	7F	00	29	DCBN CLR STORE2+\$1
058D	96	51		LDAA DGT1ST
058F	97	28		STAA STORE2

0591	D6	52		LDAB DGT2ND
0593	27	07		BEQ DC1
0595	CE	0A	00	LDX # \$0A00
0598	DF	26		STX STORE1
059A	8D	17		BSR TOBN
059C	D6	53		DC1 LDAB DGT3RD
059E	27	07		BEQ DC2
05A0	CE	64	00	LDX # \$6400
05A3	DF	26		STX STORE1
05A5	8D	0C		BSR TOBN
05A7	D6	54		DC2 LDAB DGT4TH
05A9	27	07		BEQ DC3
05AB	CE	E8	03	LDX # \$E803
05AE	DF	26		STX STORE1
05B0	8D	01		BSR TOBN
05B2	39			DC3 RTS
05B3	CE	00	28	TOBN LDX # STORE2
05B6	BD	08	02	JSR TO1
05B9	5A			DECB
05BA	26	F7		BNE TOBN
05BC	39			RTS
05BD	A6	00		FNUM LDAA X
05BF	81	B0		CMPA # \$B0
05C1	2B	04		BMI FNUM1
05C3	80	BA		SUBA # \$BA
05C5	8B	80		ADDA # \$80
05C7	39			FNUM1 RTS
05C8	DE	5C		NWQD LDX PSOLSS
05CA	86	C0		LDAA # \$C0
05CC	C6	0B		LDAB # \$0B
05CE	A7	00		CLR1 STAA X
05D0	08			INX
05D1	5A			DECB
05D2	26	FA		BNE CLR1
05D4	D6	35		LDAB CQC
05D6	C4	07		ANDB # \$07
05D8	27	04		BEQ NWQD1
05DA	DE	5C		LDX PSOLSS
05DC	8D	31		BSR LOCSET
05DE	96	35		NWQD1 LDAA CQC
05E0	BD	05	12	JSR ROTR3
05E3	16			TAB
05E4	C4	01		ANDB # \$01
05E6	27	04		BEQ NWDQ2
05E8	DE	5E		LDX PSLSS
05EA	8D	23		BSR LOCSET

05EC	96	35		NWDQ2	LDAA CQC
05EE	BD	05	11		JSR ROTR4
05F1		16			TAB
05F2	C4	03			ANDB #\$03
05F4	27	16			BEQ LLAS
05F6	DE	60			LDX PSLAS1
05F8	8D	15			BSR LOCSET
05FA	8D	10		LDAS	BSR LLAS
05FC	DE	6A			LDX PVASE1
05FE	8D	06			BSR LAS
0600	DE	6C			LDX PVASE2
0602	8D	02			BSR LAS
0604	DE	6E			LDX PVASE3
0606	A7	00		LAS	STAA X
0608	84	03			ANDA #\$03
060A	A7	01			STAA \$01,X
060C	7E	05	16	LLAS	JMP RN
060F	DF	20		LOCSET	STX PNTR1
0611	8D	F9			BSR LLAS
0613	84	3F			ANDA #\$3F
0615	8D	0B			BSR MATCH
0617	27	F8			BEQ LOCSET+\$2
0619	DE	20			LDX PNTR1
061B	A7	00			STAA X
061D		08			INX
061E		5A			DECB
061F	26	EE			BNE LOCSET
0621		39			RTS
0622	DE	5C		MATCH	LDX PSOLSS
0624	A1	00		MATCH2	CMPA X
0626	27	06			BEQ MATCH1
0628		08			INX
0629	9C	66			CPX PDVME
062B	7E	0E	82		JMP PATCH
062E		39		MATCH1	RTS
062F	96	4C		QCNT	LDAA CQLSS
0631	8A	C0			ORAA #\$C0
0633	BD	09	51		JSR ATINX1
0636	A6	00			LDAA X
0638	97	35			STAA CQC
063A		39			RTS
063B	CE	88	13	LOAD	LDX #\$8813
063E	DF	42			STX DVME
0640	CE	00	00		LDX #\$0000

0643	DF 44		STX DVSE
0645	86 0A		LDAA #\$0A
0647	97 4D		STAA NTR
0649	39		RTS
064A	CE 02 5D	TIME	LDX #\$025D
064D	BD 05 06	DONE	JSR MSG
0650	7E 09 72		JMP GALAXY
0653	CE 02 C8	LOST	LDX #\$02C8
0656	20 F5		BRA DONE
0658	CE 02 8D	WPOUT	LDX #\$028D
065B	20 F0		BRA DONE
065D	CE 04 97	EOUT	LDX #\$0497
0660	20 EB		BRA DONE
0662	DF 20	DIGPRT	STX PNTR1
0664	9F 22		STS PNTR2
0666	9E 20		LDS PNTR1
0668	DE 72		LDX PDG1ST
066A	A6 00	DGPRT1	LDAA X
066C	08		INX
066D	8A B0		ORAA #\$B0
066F	36		PSHA
0670	5A		DECB
0671	26 F7		BNE DGPRT1
0673	9E 22		LDS PNTR2
0675	39		RTS
0676	CE 01 8F	ROWSET	LDX #\$018F
0679	86 A0		LDAA #\$A0
067B	A7 00	RCLR	STAA X
067D	08		INX
067E	8C 01 A7		CPX #\$01A7
0681	26 F8		BNE RCLR
0683	17		TBA
0684	8A B0		ORAA #\$B0
0686	B7 01 8E		STAA \$018E
0689	5A		DECB
068A	DE 5A		LDX PSLOSS
068C	8D 52		BSR RWPNT
068E	26 0C		BNE STR
0690	86 BC		LDAA #\$BC
0692	A7 00		STAA X
0694	86 AA		LDAA #\$AA
0696	A7 01		STAA \$01,X
0698	86 BE		LDAA #\$BE

069A	A7 02		STAA \$02,X
069C	DE 5C		STR LDX PSOLSS
069E	DF 22		STX PNTR2
06A0	8D 3E		STR1 BSR RWPNT
06A2	26 04		BNE NXSTR
06A4	86 AA		LDAA #\$AA
06A6	A7 01		STAA \$01,X
06A8	7C 00 23		NXSTR INC PNTR2+\$1
06AB	DE 22		LDX PNTR2
06AD	9C 5E		CPX PSLSS
06AF	26 EF		BNE STR1
06B1	8D 2D		BSR RWPNT
06B3	26 0C		BNE AS
06B5	86 BE		LDAA #\$BE
06B7	A7 00		STAA X
06B9	86 B1		LDAA #\$B1
06BB	A7 01		STAA \$01,X
06BD	86 BC		LDAA #\$BC
06BF	A7 02		STAA \$02,X
06C1	DE 60		AS LDX PSLAS1
06C3	DF 22		STX PNTR2
06C5	8D 19		AS1 BSR RWPNT
06C7	26 08		BNE NXAS
06C9	86 AB		LDAA #\$AB
06CB	A7 00		STAA X
06CD	A7 01		STAA \$01,X
06CF	A7 02		STAA \$02,X
06D1	7C 00 23		NXAS INC PNTR2+\$1
06D4	DE 22		LDX PNTR2
06D6	9C 66		CPX PDVME
06D8	26 EB		BNE AS1
06DA	CE 01 8C		LDX #\$018C
06DD	7E 05 06		JMP MSG
06E0	A6 00		RWPNT LDAA X
06E2	2B 1D		BMI RWPNT1
06E4	BD 05 12		JSR ROTR3
06E7	84 07		ANDA #\$07
06E9	11		CBA
06EA	26 15		BNE RWPNT1
06EC	A6 00		LDAA X
06EE	84 07		ANDA #\$07
06F0	97 26		STAA STORE1
06F2	48		ASLA
06F3	9B 26		ADDA STORE1
06F5	8B 8F		ADDA #\$8F
06F7	7F 00 20		CLR PNTR1
06FA	7C 00 20		INC PNTR1
06FD	BD 09 54		JSR ATINX

0700	4F				CLRA
0701	39			RWPNT1	RTS
0702	CE 01	D4		QUAD	LDX #01D4
0705	DF 20				STX PNTR1
0707	CE 00	4C			LDX #CQLSS
070A	8D 06				BSR TWO
070C	CE 01	C9			LDX #01C9
070F	7E 05	06			JMP MSG
0712	A6 00			TWO	LDAA X
0714	16				TAB
0715	DE 20				LDX PNTR1
0717	BD 05	12		T1	JSR ROTR3
071A	84 07				ANDA #07
071C	8B B1				ADDA #B1
071E	A7 00				STAA X
0720	C4 07				ANDB #07
0722	CB B1				ADDB #B1
0724	E7 02				STAB \$02,X
0726	39				RTS
0727	C6 13			NTN	LDAB #13
0729	86 8D			NT1	LDAA #8D
072B	8D 0C				BSR NT3
072D	86 8A				LDAA #8A
072F	8D 08				BSR NT3
0731	86 AD			NT2	LDAA #AD
0733	8D 04				BSR NT3
0735	5A				DECB
0736	26 F9				BNE NT2
0738	39				RTS
0739	7E 0F	C0		NT3	JMP PRINT
073C	16			QDSET	TAB
073D	BD 05	11			JSR ROTR4
0740	84 03				ANDA #03
0742	8A B0				ORAA #B0
0744	A7 00				STAA X
0746	17				TBA
0747	BD 05	12			JSR ROTR3
074A	84 01				ANDA #01
074C	8A B0				ORAA #B0
074E	A7 01				STAA \$01,X
0750	C4 07				ANDB #07
0752	CA B0				ORAB #B0
0754	E7 02				STAB \$02,X
0756	39				RTS
0757	4F			CLC1	CLRA
0758	20 14				BRA LR3

075A	4F		CLC2	CLRA
075B	20	2F		BRA LR4
075D	7E	09 51	LR5	JMP ATINX1
0760	8A	C0	LRR	ORAA # \$C0
0762	16			TAB
0763	97	26		STAA STORE1
0765	C4	07		ANDB # \$07
0767	27	EE		BEQ CLC1
0769	4A			DECA
076A	8D	F1		BSR LR5
076C	A6	00		LDAA X
076E	CE	04 C9	LR3	LDX # \$04C9
0771	8D	C9		BSR QDSET
0773	96	26		LDAA STORE1
0775	8D	E6		BSR LR5
0777	A6	00		LDAA X
0779	CE	04 CF		LDX # \$04CF
077C	8D	BE		BSR QDSET
077E	96	26		LDAA STORE1
0780	16			TAB
0781	C4	07		ANDB # \$07
0783	C1	07		CMPB # \$07
0785	27	D3		BEQ CLC2
0787	4C			INCA
0788	8D	D3		BSR LR5
078A	A6	00		LDAA X
078C	CE	04 D5	LR4	LDX # \$04D5
078F	BD	07 3C		JSR QDSET
0792	CE	04 C5	LRP	LDX # \$04C5
0795	7E	05 06	LR6	JMP MSG
0798	DF	2A	ELOS	STX STORE3
079A	CE	00 2A		LDX # STORE3
079D	C6	02		LDAB # \$02
079F	BD	05 29		JSR BINDEC
07A2	CE	03 13		LDX # \$0313
07A5	C6	04		LDAB # \$04
07A7	BD	06 62		JSR DIGPRT
07AA	CE	02 FF		LDX # \$02FF
07AD	8D	E6		BSR LR6
07AF	DE	2A		LDX STORE3
07B1	DF	26		STX STORE1
07B3	8D	23	ELS1	BSR CKSD
07B5	24	32		BCC FMSD
07B7	DE	44	SDO1	LDX DVSE
07B9	DF	26		STX STORE1
07BB	8D	2C		BSR FMSD

07BD	8D 41		BSR TOMN
07BF	DE 2A		LDX STORE3
07C1	DF 26		STX STORE1
07C3	8D 17		SDO BSR CKMN
07C5	25 53		BCS EOUT1
07C7	8D 24		BSR FMMN
07C9	CE 03	15	LDX #\$0315
07CC	8D C7		BSR LR6
07CE	C6 02		LDAB #\$02
07D0	8D 3D		BSR DVD
07D2	8D 08		BSR CKMN
07D4	25 44		BCS EOUT1
07D6	20 15		BRA FMMN
07D8	DE 68		CKSD LDX PDVSE
07DA	20 02		BRA CK1
07DC	DE 66		CKMN LDX PDVME
07DE	A6 01		CK1 LDAA \$01,X
07E0	91 27		CMPA STORE1+\$1
07E2	26 35		BNE ELOS1
07E4	A6 00		CK2 LDAA X
07E6	91 26		CMPA STORE1
07E8	39		RTS
07E9	DE 68		FMSD LDX PDVSE
07EB	20 02		BRA FM1
07ED	DE 66		FMMN LDX PDVME
07EF	A6 00		FM1 LDAA X
07F1	90 26		SUBA STORE1
07F3	A7 00		STAA X
07F5	A6 01		LDAA \$01,X
07F7	92 27		SBCA STORE1+\$1
07F9	A7 01		STAA \$01,X
07FB	39		RTS
07FC	DE 68		TOSD LDX PDVSE
07FE	20 02		BRA TO1
0800	DE 66		TOMN LDX PDVME
0802	A6 00		TO1 LDAA X
0804	9B 26		ADDA STORE1
0806	A7 00		STAA X
0808	A6 01		LDAA \$01,X
080A	99 27		ADCA STORE1+\$1
080C	A7 01		STAA \$01,X
080E	39		RTS

080F	5D			DVD	TSTB
0810	76	00	27		ROR STORE1+\$1
0813	76	00	26		ROR STORE1
0816	5A				DECB
0817	26	F6			BNE DVD
0819	39			ELOS1	RTS
081A	7E	06	5D	EOUT1	JMP EOUT
081D	8D	BD		ELOM	BSR CKMN
081F	24	CC			BCC FMMN
0821	DE	26			LDX STORE1
0823	DF	2A			STX STORE3
0825	7E	07	B7		JMP SDO1
0828	DE	74		EIN	LDX PDG5TH
082A	6F	00			CLR X
082C	BD	0F	80		JSR INPUT
082F	81	AD			CMPA # \$AD
0831	26	05			BNE EN2
0833	A7	00			STAA X
0835	BD	0F	80	EN1	JSR INPUT
0838	09			EN2	DEX
0839	A7	00			STAA X
083B	BD	05	BD		JSR FNUM
083E	2B	0A			BMI EIN1
0840	A6	00			LDAA X
0842	84	0F			ANDA # \$0F
0844	A7	00			STAA X
0846	9C	72			CPX PDG1ST
0848	26	EB			BNE EN1
084A	39			EIN1	RTS
084B	86	C0		DLET	LDAA # \$C0
084D	A7	00			STAA X
084F	DF	22			STX PNTR2
0851	96	4C			LDAA CQLSS
0853	8B	C0			ADDA # \$C0
0855	BD	09	51		JSR ATINX1
0858	DF	24			STX PNTR3
085A	DE	22			LDX PNTR2
085C	BD	09	59		JSR COMPAR
085F	26	15			BNE DLAS
0861	DE	24			LDX PNTR3
0863	A6	00			LDAA X
0865	84	37			ANDA # \$37
0867	A7	00			STAA X
0869	97	35			STAA CQC
086B	7A	00	4E		DEC NSS
086E	26	1B			BNE DLET1
0870	CE	04	DB		LDX # \$04DB
0873	7E	05	06		JMP MSG

0876	DE	24		DLAS	LDX PNTR3
0878	A6	00			LDAA X
087A	80	10			SUBA # \$10
087C	A7	00			STAA X
087E	97	35			STAA CQC
0880	7A	00	4F		DEC NAS
0883	26	06			BNE DLET1
0885	CE	03	D4		LDX # \$03D4
0888	7E	06	4D		JMP DONE
088B	39			DLET1	RTS
088C	BD	0F	80	DRCT	JSR INPUT
088F	81	B1			CMPA # \$B1
0891	25	25			BCS ZRET
0893	81	B9			CMPA # \$B9
0895	24	21			BCC ZRET
0897	84	0F			ANDA # \$0F
0899	48				ASLA
089A	16				TAB
089B	86	AE			LDAA # \$AE
089D	BD	0F	C0		JSR PRINT
08A0	BD	0F	80		JSR INPUT
08A3	81	B0			CMPA # \$B0
08A5	27	04			BEQ CR1
08A7	81	B5			CMPA # \$B5
08A9	26	0D			BNE ZRET
08AB	84	01		CR1	ANDA # \$01
08AD	1B				ABA
08AE	48				ASLA
08AF	80	04			SUBA # \$04
08B1	97	21			STAA PNTR1+\$1
08B3	4F				CLRA
08B4	97	20			STAA PNTR1
08B6	4C				INCA
08B7	39				RTS
08B8	4F			ZRET	CLRA
08B9	39				RTS
08BA	9F	22		ACTV	STS PNTR2
08BC	9E	58			LDS PSTR51
08BE	96	36			LDAA SLOSS
08C0	16				TAB
08C1	C4	07			ANDB # \$07
08C3	58				ASLB
08C4	37				PSHB
08C5	84	38			ANDA # \$38
08C7	44				LSRA
08C8	44				LSRA
08C9	36				PSHA

08CA	DE	20			LDX PNTR1
08CC	A6	00			LDAA X
08CE	36				PSHA
08CF	A6	01			LDAA \$01,X
08D1	36				PSHA
08D2	9E	22			LDS PNTR2
08D4	39				RTS
08D5	7F	00	32	TRK	CLR CF
08D8	96	2F			LDAA STORE5+\$1
08DA	9B	2D			ADDA STORE4+\$1
08DC	97	2F			STAA STORE5+\$1
08DE	2A	12			BPL NOBK
08E0	84	0F			ANDA #\$0F
08E2	97	2F			STAA STORE5+\$1
08E4	7C	00	32		INC CF
08E7	96	4C			LDAA CQLSS
08E9	84	07			ANDA #\$07
08EB	27	56			BEQ TRK1
08ED	7A	00	4C		DEC CQLSS
08F0	20	17			BRA RMV
08F2	81	10		NOBK	CMPA #\$10
08F4	25	13			BCS RMV
08F6	84	0F			ANDA #\$0F
08F8	97	2F			STAA STORE5+\$1
08FA	7C	00	32		INC CF
08FD	96	4C			LDAA CQLSS
08FF	84	07			ANDA #\$07
0901	4C				INCA
0902	81	08			CMPA #\$08
0904	27	3D			BEQ TRK1
0906	7C	00	4C		INC CQLSS
0909	96	2E		RMV	LDAA STORE5
090B	9B	2C			ADDA STORE4
090D	97	2E			STAA STORE5
090F	2A	14			BPL NOUP
0911	84	0F			ANDA #\$0F
0913	97	2E			STAA STORE5
0915	7C	00	32		INC CF
0918	96	4C			LDAA CQLSS
091A	16				TAB
091B	84	38			ANDA #\$38
091D	27	24			BEQ TRK1
091F	C0	08			SUBB #\$08
0921	D7	4C			STAB CQLSS
0923	20	1A			BRA CKX

0925	81	10	NOUP	CMPA # \$10
0927	25	16		BCS CKX
0929	84	0F		ANDA # \$0F
092B	97	2E		STAA STORE5
092D	7C	00	32	INC CF
0930	96	4C		LDAA CQLSS
0932	16			TAB
0933	84	38		ANDA # \$38
0935	8B	08		ADDA # \$08
0937	81	40		CMPA # \$40
0939	27	08		BEQ TRK1
093B	CB	08		ADDB # \$08
093D	D7	4C		STAB CQLSS
093F	26	02	CKX	BNE TRK1
0941	86	01		LDAA # \$01
0943	39		TRK1	RTS
0944	96	2F	RWCM	LDAA STORE5+\$1
0946	44			LSRA
0947	84	07		ANDA # \$07
0949	D6	2E		LDAB STORE5
094B	58			ASLB
094C	58			ASLB
094D	C4	38		ANDB # \$38
094F	1B			ABA
0950	39			RTS
0951	7F	00	20	ATINX1 CLR PNTR1
0954	97	21		ATINX STAA PNTR1+\$1
0956	DE	20		LDX PNTR1
0958	39			RTS
0959	DF	20	COMPAR	STX PNTR1
095B	96	21		LDAA PNTR1+\$1
095D	81	3E		CMPA # \$3E
095F	39			RTS
0960	96	35	WASTE	LDAA CQC
0962	84	30		ANDA # \$30
0964	27	01		BEQ WASTE1
0966	39			RTS
0967	33		WASTE1	PULB
0968	33			PULB
0969	CE	04	79	LDX # \$0479
096C	BD	05	06	JSR MSG
096F	7E	0B	39	JMP CMND
0972	CE	01	00	START LDX # \$0100
0975	BD	05	06	JSR MSG
0978	BD	05	16	JSR RN

097B	BD 0F 80		JSR INPUT
097E	97 34		STAA RNM+\$1
0980	81 CE		CMPA # \$CE
0982	26 09		BNE OVER
0984	CE 04 E2		LDX # \$04E2
0987	BD 05 06		JSR MSG
098A	01		NOP
098B	01		NOP
098C	01		NOP
098D	C6 C0	OVER	LDAB # \$C0
098F	D7 26		STAB STORE1
0991	BD 05 16	GLXSET	JSR RN
0994	84 7F		ANDA # \$7F
0996	C6 0F		LDAB # \$0F
0998	D7 20		STAB PNTR1
099A	BD 09 54		JSR ATINX
099D	E6 00		LDAB X
099F	96 26		LDAA STORE1
09A1	BD 09 51		JSR ATINX1
09A4	E7 00		STAB X
09A6	7C 00 26		INC STORE1
09A9	26 E6		BNE GLXSET
09AB	7F 00 4E	GLXCK	CLR NSS
09AE	7F 00 4F		CLR NAS
09B1	CE 00 C0		LDX # \$00C0
09B4	A6 00	GLXCK1	LDAA X
09B6	16		TAB
09B7	84 08		ANDA # \$08
09B9	9B 4E		ADDA NSS
09BB	97 4E		STAA NSS
09BD	C4 30		ANDB # \$30
09BF	54		LSRB
09C0	54		LSRB
09C1	DB 4F		ADDB NAS
09C3	D7 4F		STAB NAS
09C5	08		INX
09C6	8C 01 00		CPX # \$0100
09C9	26 E9		BNE GLXCK1
09CB	96 4E		LDAA NSS
09CD	44		LSRA
09CE	44		LSRA
09CF	44		LSRA
09D0	97 4E		STAA NSS
09D2	81 07		CMPA # \$07
09D4	2A 0A		BPL SSPLS

09D6	81 02		CMPA # \$02
09D8	2A 33		BPL CAS
09DA	C6 08	SSMNS	LDAB # \$08
09DC	D7 26		STAB STORE1
09DE	20 1F		BRA MNS
09E0	C6 F7	SSPLS	LDAB # \$F7
09E2	D7 26		STAB STORE1
09E4	20 04		BRA PLS
09E6	C6 CF	ASPLS	LDAB # \$CF
09E8	D7 26		STAB STORE1
09EA	BD 05 16	PLS	JSR RN
09ED	8A C0		ORAA # \$C0
09EF	BD 09 51		JSR ATINX1
09F2	96 26		LDAA STORE1
09F4	A4 00		ANDA X
09F6	A7 00	PLS1	STAA X
09F8	7E 09 AB		JMP GLXCK
09FB	C6 10	ASMNS	LDAB # \$10
09FD	D7 26		STAB STORE1
09FF	BD 05 16	MNS	JSR RN
0A02	8A C0		ORAA # \$C0
0A04	BD 09 51		JSR ATINX1
0A07	96 26		LDAA STORE1
0A09	AA 00		ORAA X
0A0B	20 E9		BRA PLS1
0A0D	96 4F	CAS	LDAA NAS
0A0F	44		LSRA
0A10	44		LSRA
0A11	97 4F		STAA NAS
0A13	81 20		CMPA # \$20
0A15	2A CF		BPL ASPLS
0A17	81 0A		CMPA # \$0A
0A19	2B E0		BMI ASMNS
0A1B	86 05		LDAA # \$05
0A1D	9B 4F		ADDA NAS
0A1F	97 50		STAA NSR
0A21	CE 00 50		LDX # NSR
0A24	C6 01		LDAB # \$01
0A26	BD 05 29		JSR BINDEC
0A29	CE 01 4E		LDX # \$014E
0A2C	C6 02		LDAB # \$02
0A2E	BD 06 62		JSR DIGPRT
0A31	CE 00 4F		LDX # NAS
0A34	C6 01		LDAB # \$01

0A36	BD	05	29		JSR BINDEC
0A39	CE	01	3C		LDX # \$013C
0A3C	C6	02			LDAB # \$02
0A3E	BD	06	62		JSR DIGPRT
0A41	96	4E			LDAA NSS
0A43	8A	B0			ORAA # \$B0
0A45	B7	01	5F		STAA \$015F
0A48	CE	01	28		LDX # \$0128
0A4B	BD	05	06		JSR MSG
0A4E	BD	05	16		JSR RN
0A51	84	3F			ANDA # \$3F
0A53	97	4C			STAA CQLSS
0A55	BD	06	2F		JSR QCNT
0A58	BD	06	3B		JSR LOAD
0A5B	BD	05	C8		JSR NWQD
0A5E	DE	5A			LDX PSLOSS
0A60	C6	01			LDAB # \$01
0A62	BD	06	0F		JSR LOCSET
0A65	CE	01	70	SRSCN	LDX # \$0170
0A68	BD	05	06		JSR MSG
0A6B	C6	01			LDAB # \$01
0A6D	BD	06	76		JSR ROWSET
0A70	86	32			LDAA # \$32
0A72	90	50			SUBA NSR
0A74	97	26			STAA STORE1
0A76	DE	56			LDX PSTR1
0A78	C6	01			LDAB # \$01
0A7A	BD	05	29		JSR BINDEC
0A7D	CE	01	B6		LDX # \$01B6
0A80	C6	02			LDAB # \$02
0A82	BD	06	62		JSR DIGPRT
0A85	CE	01	A8		LDX # \$01A8
0A88	8D	24			BSR SRSCN1
0A8A	C6	02			LDAB # \$02
0A8C	BD	06	76		JSR ROWSET
0A8F	96	35			LDAA CQC
0A91	CE	01	C3		LDX # \$01C3
0A94	84	30			ANDA # \$30
0A96	26	19			BNE RED
0A98	86	C7			LDAA # \$C7
0A9A	A7	00			STAA X
0A9C	86	D2			LDAA # \$D2
0A9E	A7	01			STAA \$01,X
0AA0	86	C5			LDAA # \$C5
0AA2	A7	02			STAA \$02,X
0AA4	86	C5			LDAA # \$C5
0AA6	A7	03			STAA \$03,X
0AA8	86	CE			LDAA # \$CE
0AAA	A7	04			STAA \$04,X

0AAC	20 11		BRA CND
0AAE	7E 05 06	SRSCN1	JMP MSG
0AB1	86 D2	RED	LDAA #D2
0AB3	A7 00		STAA X
0AB5	86 C5		LDAA #C5
0AB7	A7 01		STAA \$01,X
0AB9	86 C4		LDAA #C4
0ABB	A7 02		STAA \$02,X
0ABD	6F 03		CLR \$03,X
0ABF	CE 01 B8	CND	LDX #01B8
0AC2	8D EA		BSR SRSCN1
0AC4	C6 03		LDAB #03
0AC6	BD 06 76		JSR ROWSET
0AC9	BD 07 02		JSR QUAD
0ACC	C6 04		LDAB #04
0ACE	BD 06 76		JSR ROWSET
0AD1	CE 01 E3		LDX #01E3
0AD4	DF 20		STX PNTR1
0AD6	DE 5A		LDX PSLOSS
0AD8	BD 07 12		JSR TWO
0ADB	CE 01 D8		LDX #01D8
0ADE	8D CE		BSR SRSCN1
0AE0	C6 05		LDAB #05
0AE2	BD 06 76		JSR ROWSET
0AE5	DE 66		LDX PDVME
0AE7	C6 02		LDAB #02
0AE9	BD 05 29		JSR BINDEC
0AEC	CE 01 F5		LDX #01F5
0AEF	C6 04		LDAB #04
0AF1	BD 06 62		JSR DIGPRT
0AF4	CE 01 E7		LDX #01E7
0AF7	8D B5		BSR SRSCN1
0AF9	C6 06		LDAB #06
0AFB	BD 06 76		JSR ROWSET
0AFE	CE 00 4D		LDX #NTR
0B01	C6 01		LDAB #01
0B03	BD 05 29		JSR BINDEC
0B06	CE 02 03		LDX #0203
0B09	C6 02		LDAB #02
0B0B	BD 06 62		JSR DIGPRT
0B0E	CE 01 F7		LDX #01F7
0B11	BD 0A AE		JSR SRSCN1
0B14	C6 07		LDAB #07
0B16	BD 06 76		JSR ROWSET
0B19	DE 68		LDX PDVSE

0B1B	C6 02		LDAB # \$02
0B1D	BD 05 29		JSR BINDEC
0B20	CE 02 13		LDX # \$0213
0B23	C6 04		LDAB # \$04
0B25	BD 06 62		JSR DIGPRT
0B28	CE 02 05		LDX # \$0205
0B2B	BD 05 06		JSR MSG
0B2E	C6 08		LDAB # \$08
0B30	BD 06 76		JSR ROWSET
0B33	CE 01 70		LDX # \$0170
0B36	BD 05 06		JSR MSG
0B39	CE 0A 00	CMND	LDX # \$0A00
0B3C	DF 26		STX STORE1
0B3E	BD 08 1D		JSR ELOM
0B41	7A 00 34		DEC RNM+\$1
0B44	CE 02 15	CMD	LDX # \$0215
0B47	BD 05 06		JSR MSG
0B4A	BD 0F 80		JSR INPUT
0B4D	81 B0		CMPA # \$B0
0B4F	26 03		BNE NCRSE
0B51	7E 0C 4C		JMP CRSE
0B54	81 B1	NCRSE	CMPA # \$B1
0B56	26 03		BNE NSRSCN
0B58	7E 0A 65		JMP SRSCN
0B5B	81 B2	NSRSCN	CMPA # \$B2
0B5D	26 03		BNE NLRSCN
0B5F	7E 0B 7E		JMP LRSCN
0B62	81 B3	NLRSCN	CMPA # \$B3
0B64	26 03		BNE NGXPRT
0B66	7E 0B D3		JMP GXPRT
0B69	81 B4	NGXPRT	CMPA # \$B4
0B6B	26 03		BNE NSHEN
0B6D	7E 0C 13		JMP SHEN
0B70	81 B5	NSHEN	CMPA # \$B5
0B72	26 03		BNE NPHSR
0B74	7E 0D B8		JMP PHSR
0B77	81 B6	NPHSR	CMPA # \$B6
0B79	26 C9		BNE CMD
0B7B	7E 0D 32		JMP TRPD
0B7E	CE 02 4D	LRSCN	LDX # \$024D
0B81	BD 05 06		JSR MSG
0B84	BD 07 02		JSR QUAD
0B87	8D 25		BSR LRSCN1
0B89	96 4C		LDAA CQLSS
0B8B	16		TAB
0B8C	C4 38		ANDB # \$38

0B8E	27 21		BEQ RWC1
0B90	80 08		SUBA # \$08
0B92	BD 07 60		JSR LRR
0B95	8D 17	LR1	BSR LRSCN1
0B97	96 4C		LDAA CQLSS
0B99	BD 07 60		JSR LRR
0B9C	8D 10		BSR LRSCN1
0B9E	96 4C		LDAA CQLSS
0BA0	81 38		CMPA # \$38
0BA2	24 12		BCC RWC2
0BA4	8B 08		ADDA # \$08
0BA6	BD 07 60		JSR LRR
0BA9	8D 03	LR2	BSR LRSCN1
0BAB	7E 0B 39		JMP CMND
0BAE	7E 07 27	LRSCN1	JMP NTN
0BB1	8D 08	RWC1	BSR RWC
0BB3	7E 0B 95		JMP LR1
0BB6	8D 03	RWC2	BSR RWC
0BB8	7E 0B A9		JMP LR2
0BBB	CE 04 C9	RWC	LDX # \$04C9
0BBE	4F		CLRA
0BBF	BD 07 3C		JSR QDSET
0BC2	CE 04 CF		LDX # \$04CF
0BC5	4F		CLRA
0BC6	BD 07 3C		JSR QDSET
0BC9	CE 04 D5		LDX # \$04D5
0BCC	4F		CLRA
0BCD	BD 07 3C		JSR QDSET
0BD0	7E 07 92		JMP LRP
0BD3	CE 04 22	GXPRT	LDX # \$0422
0BD6	BD 05 06		JSR MSG
0BD9	C6 31		LDAB # \$31
0BDB	BD 07 29		JSR NT1
0BDE	CE 00 C0		LDX # \$00C0
0BE1	DF 20		STX PNTR1
0BE3	CE 00 84	GL1	LDX # \$0084
0BE6	DF 22		STX PNTR2
0BE8	DE 20	GL2	LDX PNTR1
0BEA	8C 01 00		CPX # \$0100
0BED	27 21		BEQ GL3
0BEF	A6 00		LDAA X
0BF1	08		INX
0BF2	DF 20		STX PNTR1
0BF4	DE 22		LDX PNTR2

0BF6	BD 07 3C		JSR QDSET
0BF9	86 06		LDAA #\$06
0BFB	9B 23		ADDA PNTR2+\$1
0BFD	97 23		STAA PNTR2+\$1
0BFF	81 B4		CMPA #\$B4
0C01	26 E5		BNE GL2
0C03	CE 00 80		LDX #\$0080
0C06	BD 05 06		JSR MSG
0C09	C6 31		LDAB #\$31
0C0B	BD 07 29		JSR NT1
0C0E	20 D3		BRA GL1
0C10	7E 0B 39	GL3	JMP CMND
0C13	CE 03 30	SHEN	LDX #\$0330
0C16	BD 05 06		JSR MSG
0C19	BD 08 28		JSR EIN
0C1C	2B F5		BMI SHEN
0C1E	BD 05 8A		JSR DCBN
0C21	DE 28		LDX STORE2
0C23	DF 26		STX STORE1
0C25	96 55		LDAA DGT5TH
0C27	27 0D		BEQ POS
0C29	BD 07 D8		JSR CKSD
0C2C	25 15		BCS NE
0C2E	BD 07 E9		JSR FMDS
0C31	BD 08 00		JSR TOMN
0C34	20 13		BRA SHEN1
0C36	BD 07 DC	POS	JSR CKMN
0C39	25 08		BCS NE
0C3B	BD 07 ED		JSR FMMN
0C3E	BD 07 FC		JSR TOSD
0C41	20 06		BRA SHEN1
0C43	CE 03 4C	NE	LDX #\$034C
0C46	BD 05 06		JSR MSG
0C49	7E 0B 39	SHEN1	JMP CMND
0C4C	CE 02 20	CRSE	LDX #\$0220
0C4F	BD 05 06		JSR MSG
0C52	BD 08 8C		JSR DRCT
0C55	27 F5		BEQ CRSE
0C57	CE 02 33	WRP	LDX #\$0233
0C5A	BD 05 06		JSR MSG
0C5D	BD 0F 80		JSR INPUT
0C60	81 B0		CMPA #\$B0
0C62	25 F3		BCS WRP

0C64	81	B8		CMPA #B8
0C66	24	EF		BCC WRP
0C68	84	07		ANDA #07
0C6A	48			ASLA
0C6B	48			ASLA
0C6C	48			ASLA
0C6D	16			TAB
0C6E	86	AE		LDAA #SAE
0C70	BD	0F	C0	JSR PRINT
0C73	BD	0F	80	JSR INPUT
0C76	81	B0		CMPA #B0
0C78	25	DD		BCS WRP
0C7A	81	B8		CMPA #B8
0C7C	24	D9		BCC WRP
0C7E	84	07		ANDA #07
0C80	1B			ABA
0C81	27	D4		BEQ WRP
0C83	97	30		STAA CNTR
0C85	BD	08	BA	JSR ACTV
0C88	7F	00	31	CLR CI
0C8B	BD	08	D5	MOV JSR TRK
0C8E	26	03		BNE MOV1
0C90	7E	06	53	JMP LOST
0C93	96	32		MOV1 LDAA CF
0C95	27	10		BEQ CLSN
0C97	97	31		STAA CI
0C99	CE	19	00	LDX #1900
0C9C	DF	26		STX STORE1
0C9E	BD	08	1D	JSR ELOM
0CA1	BD	06	2F	JSR QCNT
0CA4	BD	05	C8	JSR NWQD
0CA7	BD	09	44	CLSN JSR RWCM
0CAA	BD	06	22	JSR MATCH
0CAD	26	0E		BNE MVDN
0CAF	BD	09	59	JSR COMPAR
0CB2	27	2D		BEQ SSOUT
0CB4	24	43		BCC ASOUT
0CB6	96	31		LDAA CI
0CB8	26	03		BNE MVDN
0CBA	7E	06	58	JMP WPOUT
0CBD	7A	00	30	MVDN DEC CNTR
0CC0	26	C9		BNE MOV
0CC2	96	31		LDAA CI
0CC4	27	08		BEQ NOX
0CC6	7A	00	50	DEC NSR
0CC9	26	03		BNE NOX

0CCB	7E 06 4A		JMP TIME
0CCE	BD 09 44	NOX	JSR RWCM
0CD1	97 36		STAA SLOSS
0CD3	BD 06 22		JSR MATCH
0CD6	26 03		BNE NOX1
0CD8	BD 0D 0D		JSR CHNG
0CDB	BD 0D 12	NOX1	JSR DKED
0CDE	7E 0A 65		JMP SRSCN
0CE1	96 31	SSOUT	LDAA CI
0CE3	26 D8		BNE MVDN
0CE5	BD 08 4B		JSR DLET
0CE8	CE 03 BA		LDX # \$03BA
0CEB	BD 05 06		JSR MSG
0CEE	CE 58 02		LDX # \$5802
0CF1	DF 26		STX STORE1
0CF3	BD 07 98	SSO1	JSR ELOS
0CF6	7E 0C BD		JMP MVDN
0CF9	96 31	ASOUT	LDAA CI
0CFB	26 C0		BNE MVDN
0CFD	BD 08 4B		JSR DLET
0D00	CE 03 7F		LDX # \$037F
0D03	BD 05 06		JSR MSG
0D06	CE DC 05		LDX # \$DC05
0D09	DF 26		STX STORE1
0D0B	20 E6		BRA SSO1
0D0D	C6 01	CHNG	LDAB # \$01
0D0F	7E 06 0F		JMP LOCSET
0D12	96 3E	DKED	LDAA SLSS
0D14	2A 01		BPL DKED1
0D16	39		RTS
0D17	84 38	DKED1	ANDA # \$38
0D19	D6 36		LDAB SLOSS
0D1B	C4 38		ANDB # \$38
0D1D	11		CBA
0D1E	26 0E		BNE DKED2
0D20	96 3E		LDAA SLSS
0D22	D6 36		LDAB SLOSS
0D24	CB 01		ADDB # \$01
0D26	11		CBA
0D27	27 06		BEQ DKED3
0D29	C0 02		SUBB # \$02
0D2B	11		CBA
0D2C	27 01		BEQ DKED3
0D2E	39	DKED2	RTS

0D2F	7E 06 3B	DKED3	JMP LOAD
0D32	96 4D	TRPD	LDAA NTR
0D34	27 79		BEQ NTPD
0D36	7A 00 4D		DEC NTR
0D39	CE FA 00		LDX # \$FA00
0D3C	DF 26		STX STORE1
0D3E	BD 07 DC		JSR CKMN
0D41	24 03		BCC TRPD1
0D43	7E 0C 43		JMP NE
0D46	BD 07 ED	TRPD1	JSR FMMN
0D49	CE 03 60	TR1	LDX # \$0360
0D4C	8D 41		BSR TR3
0D4E	BD 08 8C		JSR DRCT
0D51	27 F6		BEQ TR1
0D53	BD 08 BA		JSR ACTV
0D56	96 4C		LDAA CQLSS
0D58	97 30		STAA CNTR
0D5A	BD 08 D5	TR2	JSR TRK
0D5D	27 3B		BEQ QOUT
0D5F	96 32		LDAA CF
0D61	26 37		BNE QOUT
0D63	BD 09 44		JSR RWCM
0D66	16		TAB
0D67	97 26		STAA STORE1
0D69	CE 04 1E		LDX # \$041E
0D6C	BD 07 17		JSR T1
0D6F	CE 04 12		LDX # \$0412
0D72	8D 1B		BSR TR3
0D74	96 26		LDAA STORE1
0D76	BD 06 22		JSR MATCH
0D79	27 03		BEQ HIT
0D7B	7E 0D 5A		JMP TR2
0D7E	BD 09 59	HIT	JSR COMPAR
0D81	25 17		BCS QOUT
0D83	27 0D		BEQ SSTA
0D85	BD 08 4B		JSR DLET
0D88	CE 03 7F		LDX # \$037F
0D8B	8D 02		BSR TR3
0D8D	20 26		BSR CMND1
0D8F	7E 05 06	TR3	JMP MSG
0D92	BD 08 4B	SSTA	JSR DLET
0D95	CE 03 BA		LDX # \$03BA
0D98	8D F5		BSR TR3
0D9A	96 30	QOUT	LDAA CNTR

0D9C	97 4C		STAA CQLSS
0D9E	BD 09 60		JSR WASTE
0DA1	CE 03 96		LDX # \$0396
0DA4	BD 05 06		JSR MSG
0DA7	CE C8 00		LDX # \$C800
0DAA	BD 07 98		JSR ELOS
0DAD	20 06		BRA CMND1
0DAF	CE 04 B6	NTPD	LDX # \$04B6
0DB2	BD 05 06		JSR MSG
0DB5	7E 0B 39	CMND1	JMP CMND
0DB8	CE 04 33	PHSR	LDX # \$0433
0DBB	8D D2		BSR TR3
0DBD	BD 08 28		JSR EIN
0DC0	2B F6		BMI PHSR
0DC2	BD 05 8A		JSR DCBN
0DC5	DE 28		LDX STORE2
0DC7	DF 26		STX STORE1
0DC9	BD 08 1D		JSR ELOM
0DCC	BD 09 60		JSR WASTE
0DCF	BD 05 11	PHS1	JSR ROTR4
0DD2	80 01		SUBA # \$01
0DD4	27 04		BEQ PH1
0DD6	16		TAB
0DD7	BD 08 0F		JSR DVD
0DDA	DE 26	PH1	LDX STORE1
0DDC	DF 2C		STX STORE4
0DDE	DE 6A		LDX PVASE1
0DE0	DF 24		STX PNTR3
0DE2	DE 60		LDX PSLAS1
0DE4	BD 0D FB		JSR ASPH
0DE7	DE 6C		LDX PVASE2
0DE9	DF 24		STX PNTR3
0DEB	DE 62		LDX PSLAS2
0DED	BD 0D FB		JSR ASPH
0DF0	DE 6E		LDX PVASE3
0DF2	DF 24		STX PNTR3
0DF4	DE 64		LDX PSLAS3
0DF6	BD 0D FB		JSR ASPH
0DF9	20 BA		BRA CMND1
0DFB	DF 22	ASPH	STX PNTR2
0DFD	A6 00		LDAA X
0DFF	2A 01		BPL ASPH1
0E01	39		RTS
0E02	DE 2C	ASPH1	LDX STORE4
0E04	DF 26		STX STORE1
0E06	CE 04 65		LDX # \$0465
0E09	DF 20		STX PNTR1

0E0B	DE 22		LDX PNTR2
0E0D	BD 07 12		JSR TWO
0E10	CE 04 4E		LDX # \$044E
0E13	BD 05 06		JSR MSG
0E16	CE 00 36		LDX # \$SLOSS
0E19	8D 5C		BSR SPRC
0E1B	97 28		STAA STORE2
0E1D	D7 29		STAB STORE2+\$1
0E1F	DE 22		LDX PNTR2
0E21	8D 54		BSR SPRC
0E23	90 28		SUBA STORE2
0E25	2A 01		BPL PH2
0E27	40		NEGA
0E28	D0 29	PH2	SUBB STORE2+\$1
0E2A	2A 01		BPL PH3
0E2C	50		NEGB
0E2D	1B	PH3	ABA
0E2E	44		LSRA
0E2F	44		LSRA
0E30	84 03		ANDA # \$03
0E32	16		TAB
0E33	27 03		BEQ PH4
0E35	BD 08 0F		JSR DVD
0E38	DE 24	PH4	LDX PNTR3
0E3A	BD 07 EF		JSR FM1
0E3D	2B 2D		BMI DSTR
0E3F	26 04		BNE ALOS
0E41	6D 00		TST X
0E43	27 27		BEQ DSTR
0E45	C6 02	ALOS	LDAB # \$02
0E47	BD 05 29		JSR BINDEC
0E4A	CE 04 77		LDX # \$0477
0E4D	C6 04		LDAB # \$04
0E4F	BD 06 62		JSR DIGPRT
0E52	CE 04 6B		LDX # \$046B
0E55	BD 05 06		JSR MSG
0E58	DE 24		LDX PNTR3
0E5A	A6 00		LDAA X
0E5C	97 26		STAA STORE1
0E5E	A6 01		LDAA \$01,X
0E60	97 27		STAA STORE1+\$1
0E62	C6 02		LDAB # \$02
0E64	BD 08 0F		JSR DVD
0E67	DE 26		LDX STORE1
0E69	7E 07 98		JMP ELOS
0E6C	CE 03 CA	DSTR	LDX # \$03CA
0E6F	BD 05 06		JSR MSG
0E72	DE 22		LDX PNTR2
0E74	7E 08 4B		JMP DLET

0E77	A6 00	SPRC	LDAA X
0E79	16		TAB
0E7A	BD 05 12		JSR ROTR3
0E7D	84 07		ANDA #\$07
0E7F	C4 07		ANDB #\$07
0E81	39		RTS
0E82	26 03	PATCH	BNE PATCH1
0E84	08		INX
0E85	09		DEX
0E86	39		RTS
0E87	7E 06 24	PATCH1	JMP MATCH2

0F00	00 01 04 23 0A 03 07 00
0F08	00 1A 23 05 03 14 16 12
0F10	00 00 00 00 00 05 04 17
0F18	05 01 14 00 00 04 05 00
0F20	07 02 11 09 00 04 00 00
0F28	23 00 02 24 00 00 03 07
0F30	00 15 00 05 0E 00 02 06
0F38	15 00 03 02 13 00 34 03
0F40	07 01 00 00 00 03 15 00
0F48	00 04 00 1F 04 01 03 02
0F50	03 14 00 00 00 16 0D 00
0F58	00 04 13 03 00 00 00 14
0F60	0B 01 15 13 00 00 00 03
0F68	07 00 00 00 1D 04 00 16
0F70	00 13 15 00 00 04 06 02
0F78	03 15 00 00 16 00 27 00

0F80	BD E1 AC	INPUT	JSR \$E1AC
0F83	8A 80		ORAA #\$80
0F85	39		RTS
0FC0	36	PRINT	PSHA
0FC1	BD E1 D1		JSR \$E1D1
0FC4	32		PULA
0FC5	39		RTS

SAMPLE OF GALAXY OPERATION

For those that may still be unsure of the operation of the Galaxy game, the following sample illustrates the initial moves that may be made in a typical game. The galaxy contents are assumed to be the same as that displayed on page 1-8. All operator entries are underlined. The comments in the parentheses are included to point out various facts one should watch as a game progresses, and to explain the reasoning behind each of the moves. The Galaxy game is initiated by jumping to the address 0500 **hexadecimal**.

DO YOU WANT TO GO ON A SPACE VOYAGE? Y

YOU MUST DESTROY 22 ALIEN SHIPS IN 27 STARDATES
WITH 4 SPACE STATIONS

<pre> - 1 - - 2 - - 3 - - 4 - - 5 - - 6 - - 7 - - 8 - 1 * 2 3 + + + 4 * 5 < * > 6 7 > 1 < * 8 - 1 - - 2 - - 3 - - 4 - - 5 - - 6 - - 7 - - 8 - </pre>	<table border="0" style="width: 100%;"> <tr> <td style="width: 150px;">STARDATE</td> <td>3023</td> </tr> <tr> <td>CONDITION</td> <td>RED</td> </tr> <tr> <td>QUADRANT</td> <td>6,5</td> </tr> <tr> <td>SECTOR</td> <td>5,3</td> </tr> <tr> <td>ENERGY</td> <td>5000</td> </tr> <tr> <td>TORPEDOES</td> <td>10</td> </tr> <tr> <td>SHIELDS</td> <td>0000</td> </tr> </table>	STARDATE	3023	CONDITION	RED	QUADRANT	6,5	SECTOR	5,3	ENERGY	5000	TORPEDOES	10	SHIELDS	0000
STARDATE	3023														
CONDITION	RED														
QUADRANT	6,5														
SECTOR	5,3														
ENERGY	5000														
TORPEDOES	10														
SHIELDS	0000														

(Before attacking the alien ship, energy should be transferred to the protective shields.)

COMMAND? 4

SHIELD ENERGY TRANSFER = 1000

(The alien ship is located three columns to the right and two rows up. A torpedo trajectory of 1.5 just might make it.)

COMMAND? 6

TORPEDO TRAJECTORY: 1.5

TRACKING 4,4

TRACKING 4,5

TRACKING 3,6

ALIEN SHIP DESTROYED

(Good shot. Now, a short range scan will indicate the loss of the alien ship and amount of energy remaining. The energy consumed was 10 units for each command entered plus 250 units to fire the torpedo.)

COMMAND? 1

- 1 -- 2 -- 3 -- 4 -- 5 -- 6 -- 7 -- 8 -	
1 *	STARDATE 3023
2	CONDITION GREEN
3	QUADRANT 6,5
4 *	SECTOR 5,3
5 <*>	ENERGY 3720
6	TORPEDOES 09
7 >1< *	SHIELDS 1000
8	
- 1 -- 2 -- 3 -- 4 -- 5 -- 6 -- 7 -- 8 -	

(Before leaving this quadrant, docking with the space station will refill the energy banks and torpedo tubes.)

COMMAND? 0

COURSE (1 - 8.5)? 7.0

WARP FACTOR (0.1 - 7.7)? 0.2


```

- 1 - - 2 - - 3 - - 4 - - 5 - - 6 - - 7 - - 8 -
1          *
2
3
4      *
5
6
7      <*>>1<          *
8
- 1 - - 2 - - 3 - - 4 - - 5 - - 6 - - 7 - - 8 -

```

```

STARDATE      3023
CONDITION      GREEN
QUADRANT       6,5
SECTOR         7,3
ENERGY         5000
TORPEDOES      10
SHIELDS        0000

```

(A long range scan will display the surrounding quadrants.)

COMMAND? 2

LONG RANGE SCAN FOR QUADRANT 6,5

```

-----
1 112 1 001 1 006 1
-----
1 001 1 013 1 104 1
-----
1 203 1 007 1 004 1
-----

```

(Let's move into quadrant 7,4 to attack the two alien ships residing there. The stardate will increase by one, and the new quadrant location will be indicated. If the move is tracked one sector at a time it would be noted that two quadrant borders were crossed, resulting in the loss of 25 units of energy for each crossing.)

COMMAND? 0

COURSE (1 - 8.5)? 6.0

WARP FACTOR (0.1 - 7.7)? 1.0

```

- 1 -- 2 -- 3 -- 4 -- 5 -- 6 -- 7 -- 8 -
1
2
3      +++
4      *
5
6 +++ *
7      * <*>
8
- 1 -- 2 -- 3 -- 4 -- 5 -- 6 -- 7 -- 8 -

```

```

STARDATE      3024
CONDITION     RED
QUADRANT      7,4
SECTOR        7,3
ENERGY        4930
TORPEDOES     10
SHIELDS       0000

```

(Don't forget the shield energy before attacking.)

COMMAND? 4

SHIELD ENERGY TRANSFER = 1000

(The stars are blocking the path to both alien ships for the torpedoes. Instead of maneuvering to a position to fire a torpedo at each, a small phasor is fired to determine the size of the alien ships.)

COMMAND? 5

PHASOR ENERGY TO FIRE = 0100

ALIEN SHIP AT SECTOR 3,3: DESTROYED

ALIEN SHIP AT SECTOR 6,1: ENERGY = 0150

LOSS OF ENERGY 0037

(The alien ship at sector 3,3 was destroyed. The other alien ship fired back in retaliation. However, since its shield energy is only 150, and the distance factor (as defined on page 1 - 10) is zero, another phasor shot should take care of it.)

COMMAND? 5

PHASOR ENERGY TO FIRE = 0150

ALIEN SHIP AT SECTOR 6,1: DESTROYED

(A short range scan will provide proof that the alien ships are destroyed, and also indicate how much energy is left.)

COMMAND? 1

- 1 - - 2 - - 3 - - 4 - - 5 - - 6 - - 7 - - 8 -		
1	STARDATE	3024
2	CONDITION	GREEN
3	QUADRANT	7,4
4 *	SECTOR	7,3
5	ENERGY	3640
6 *	TORPEDOES	10
7 * <*>	SHIELDS	0963
8		
- 1 - - 2 - - 3 - - 4 - - 5 - - 6 - - 7 - - 8 -		

(The game would be continued by maneuvering about to the other quadrants in the galaxy which contain alien ships. However, one must always be aware of the amount of energy in the space ship, and the number of stardates remaining as the game progresses. Allowing either of these to run out would be as disastrous as moving out of the known galaxy or making a fatal error such as the following attempt to move to quadrant 5,4.)

COMMAND? 0

COURSE (1 - 8.5)? 3.0

WARP FACTOR (0.1 - 7.7)? 2.1

KA-BOOM, YOU CRASHED INTO A STAR.
YOUR SHIP IS DESTROYED.

NOTES

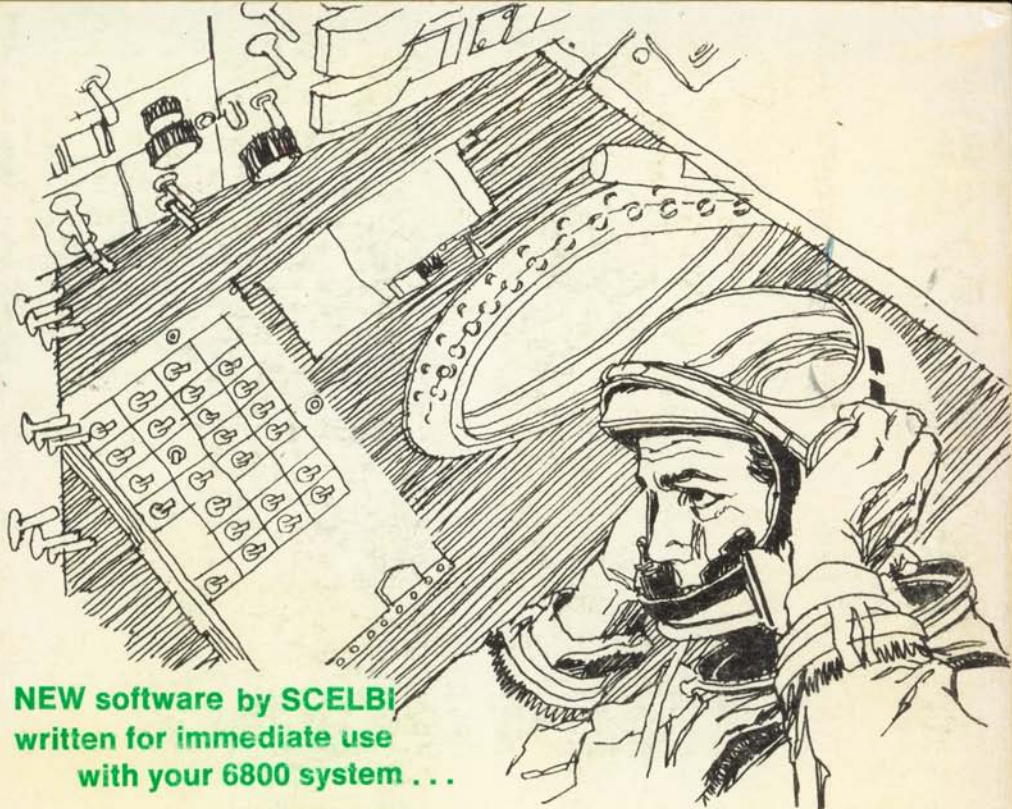
NOTES

PUBLICATIONS FROM SCALBI COMPUTER CONSULTING, INC.

AN '8080' ASSEMBLER PROGRAM.....	\$19.95
AN '8080' EDITOR PROGRAM.....	\$17.95
'8080' MONITOR ROUTINES.....	\$14.95
SCALBI'S GALAXY GAME FOR THE '8008/8080'	\$14.95
SCALBI'S GALAXY GAME FOR THE '6800'	\$14.95
SCALBI'S FIRST BOOK OF COMPUTER GAMES FOR THE '8008/8080'.....	\$14.95
SCALBI '8080' SOFTWARE GOURMET GUIDE AND COOK BOOK.....	\$ 9.95
SCALBI '6800' SOFTWARE GOURMET GUIDE AND COOK BOOK.....	\$ 9.95

**THE ABOVE PUBLICATIONS MAY BE ORDERED
DIRECTLY FROM:**

**SCALBI COMPUTER CONSULTING, INC.
1322 Rear - Boston Post Road
Milford, CT 06460**



NEW software by SCELBI
written for immediate use
with your 6800 system . . .

SCELBI'S
"GALAXY"
GAME FOR THE "6800"

Captain your own crusading starship against the logic of your "6800". Your mission: search-and-destroy a random number of alien ships. But, don't run out of time, out of fuel, out of ammunition or out of the galaxy. Your galaxy consists of 64 quadrants, subdivided into 64 sectors. Plan your mission to destroy all aliens. But, every time you move you lose a stardate and precious fuel. Don't run into a roaming star that could damage your ship! And, don't forget how much fuel your warp factor uses! Suddenly, *Condition RED! Alien in sight!* How big is he? Fire a phasor or torpedo? He's damaged or destroyed. But, you've used up valuable fuel! Does he fire back? How much fuel was used for protective shields? Be careful. You're running out of time and fuel. You get the idea. You must maneuver logically, strategically, carefully . . . to complete your mission. Here's the multidimensional computer game you've asked for. Using the original manufacturer's recommended mnemonics and assembly format, with hexadecimal notations, you've got a total book form program in machine language, for 4K memory, with flow charts, illustrations and more.



SCELBI COMPUTER
CONSULTING INC.

1322 Rear Boston Post Road, Milford, CT 06460 • Telephone: 203/874-1573